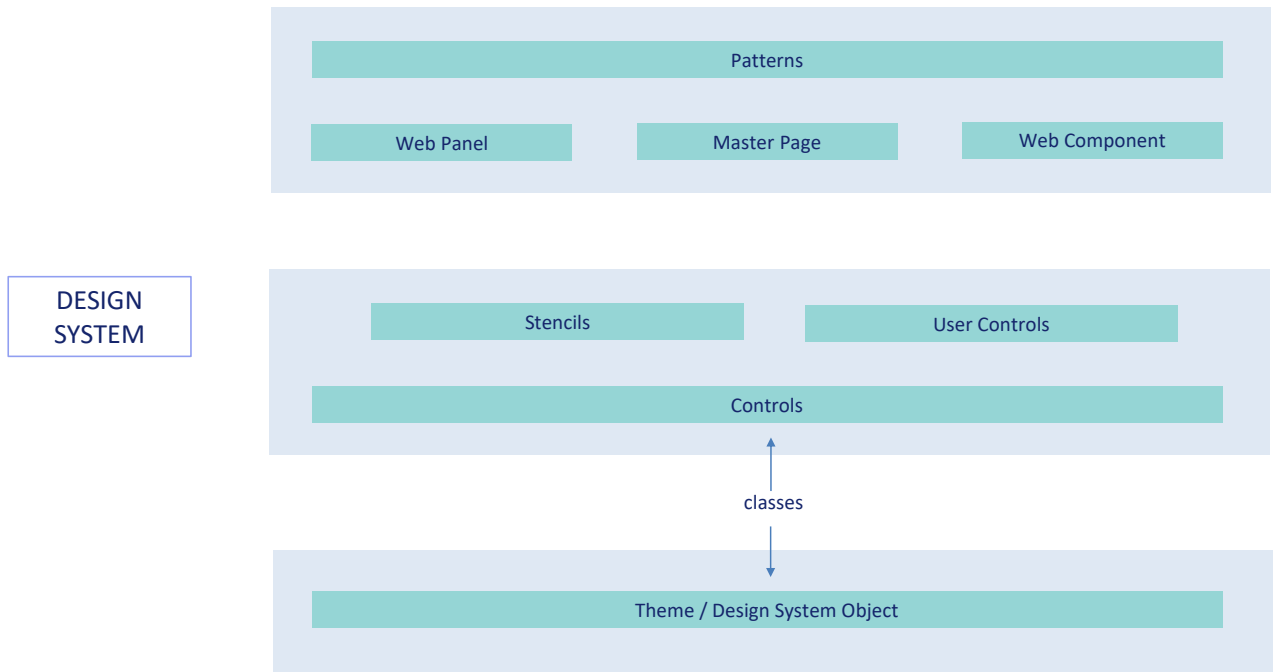


Design System in GeneXus

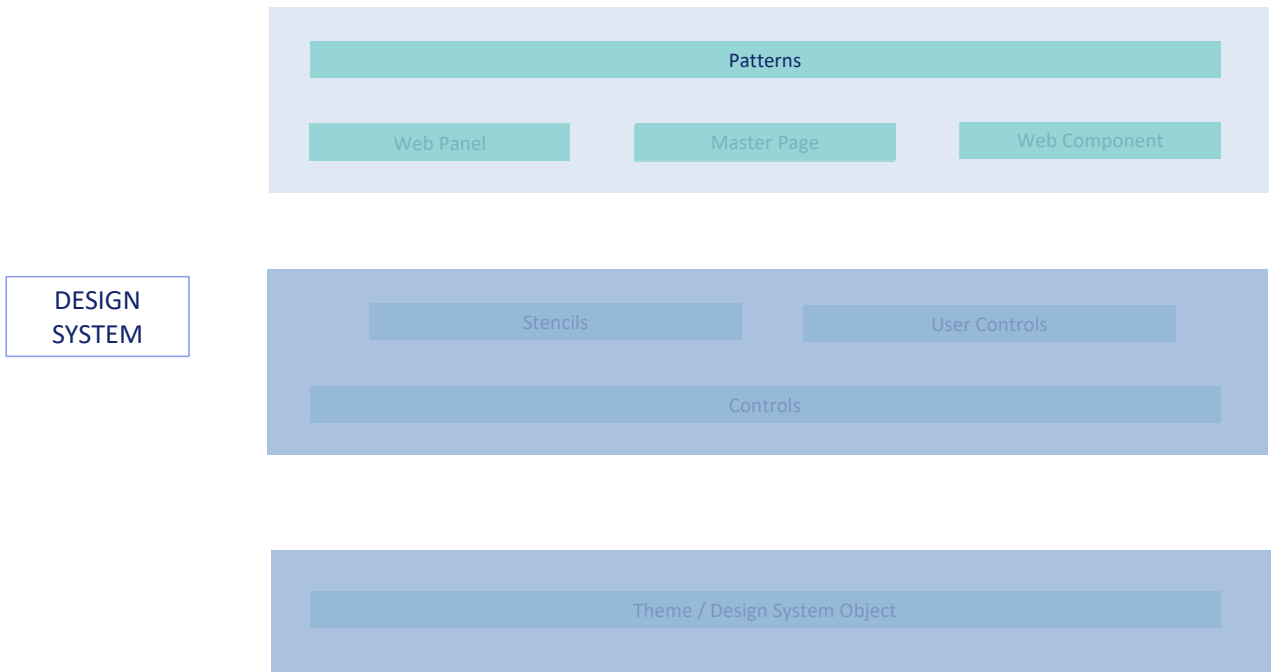
Overview

GeneXus™



In GeneXus, which players take part in modeling the Design System of an application?

Here we will briefly describe them. After this course, you will be able to learn more about them if you want to focus on the User Interface of your application.



Using a top-down approach:

We have Patterns, such as the Work With of a transaction, which create Web Panel objects, among other things. These panels already include part of the Design System.

For example, actions have a common aesthetic, with a color that is also used in the buttons of the transaction or Web panel to display the information of an item. There is also a common header that all pages share by default. Not only those of the Work With, but also those of the Web Panels we created from scratch. To avoid repeating this common header in all Web Panels, there is a Master Page object.

Each web panel will be loaded inside the ContentPlaceHolder control of its Master Page. And here is that header we were looking at. When the Web Panel is executed, its Master Page is also executed and the page viewed in the Browser is the combination of both layouts, according to what the Master Page indicates.

GeneXus Developer Menu Attractions

apps5.genexus.com/ld7a09fb2c7edf01b0019064c3370a693f/wwattraction.aspx

Apps GXSync Web Angular application... Travel Agency Sam... Reading list

Travel Agency

Default Design System: CARMINE

Recents Attraction — Christ the Redemmer — WWAttractions From... — Attractions

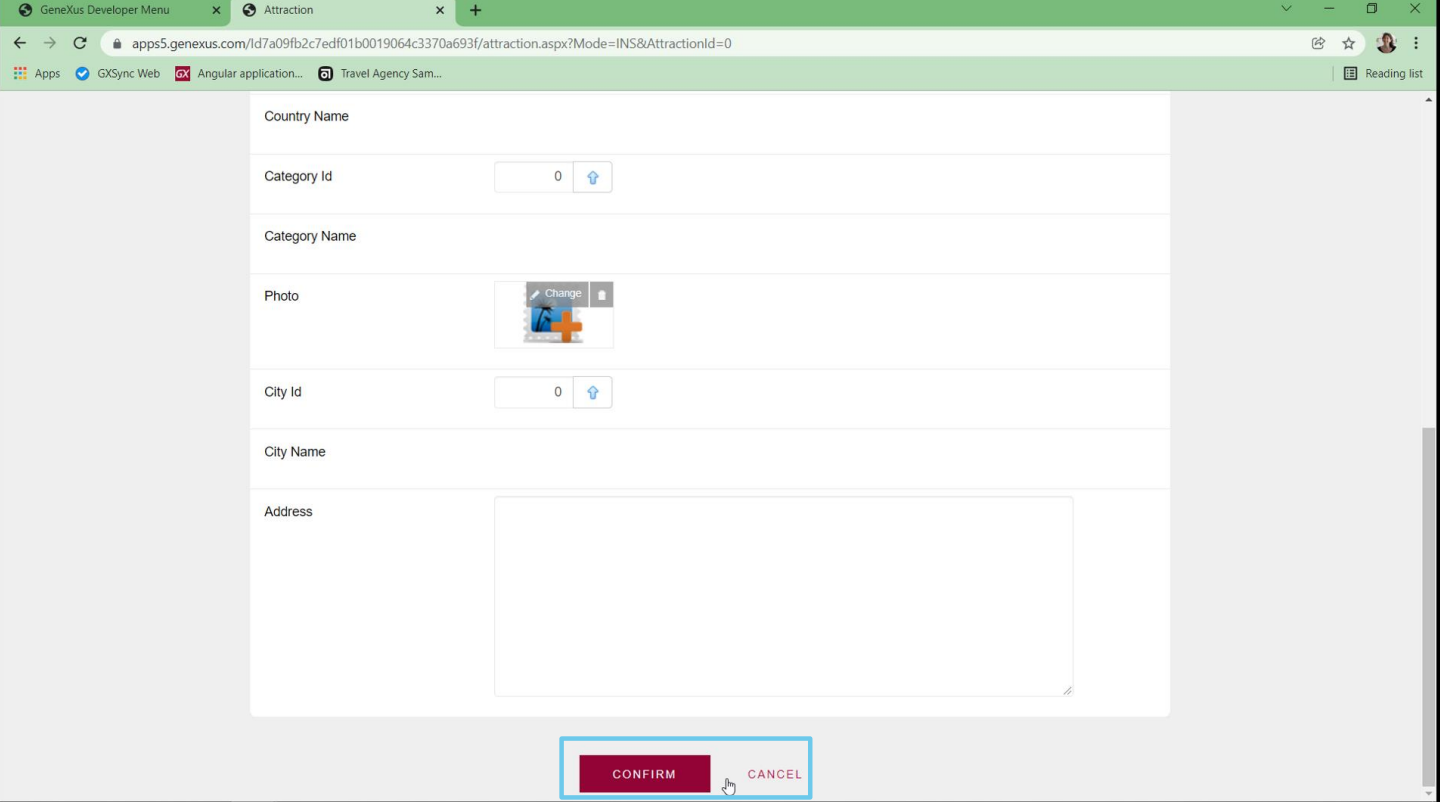
HIDE FILTERS Attractions Name + INSERT

Ordered By : Name

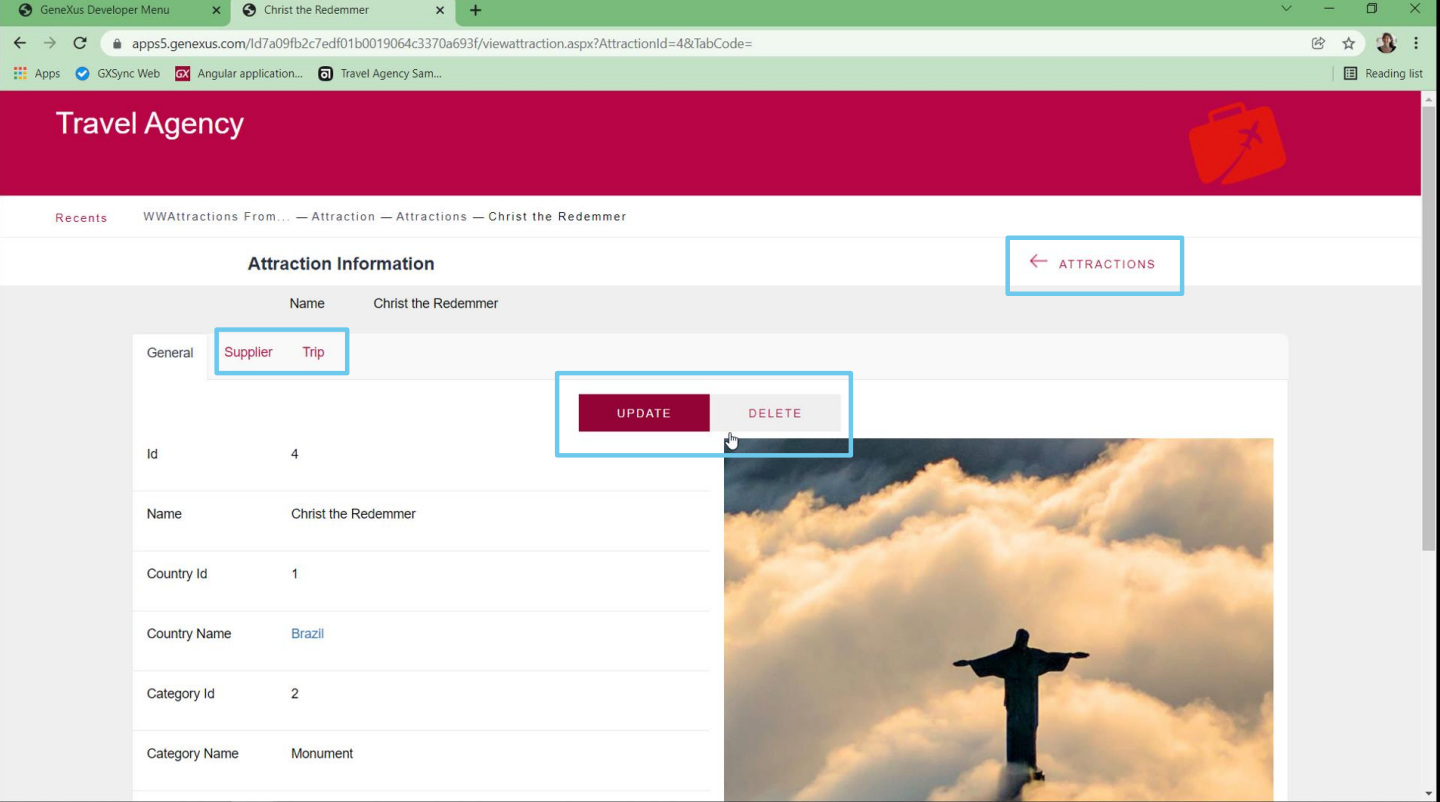
COUNTRY NAME

Id	Name	Country Name	Category Name	Photo	City Name	UPDATE	DELETE
4	Christ the Redemmer	Brazil	Monument		Rio de Janeiro	UPDATE	DELETE
12	Cinque Terre	Italy	Tourist site		Maranola	UPDATE	DELETE
3	Eiffel Tower	France	Monument		Paris	UPDATE	DELETE
7	Forbidden city	China	Tourist site		Beijing	UPDATE	DELETE
8	Glenfinnan Viaduct	Scotland	Tourist site		Glenfinnan	UPDATE	DELETE
13	London Bridges	United States	Tourist site		San Francisco	UPDATE	DELETE
44	London Tower	England	Monument		London	UPDATE	DELETE

For example, we see that the actions have a common aesthetic, with a color that is repeated as well....

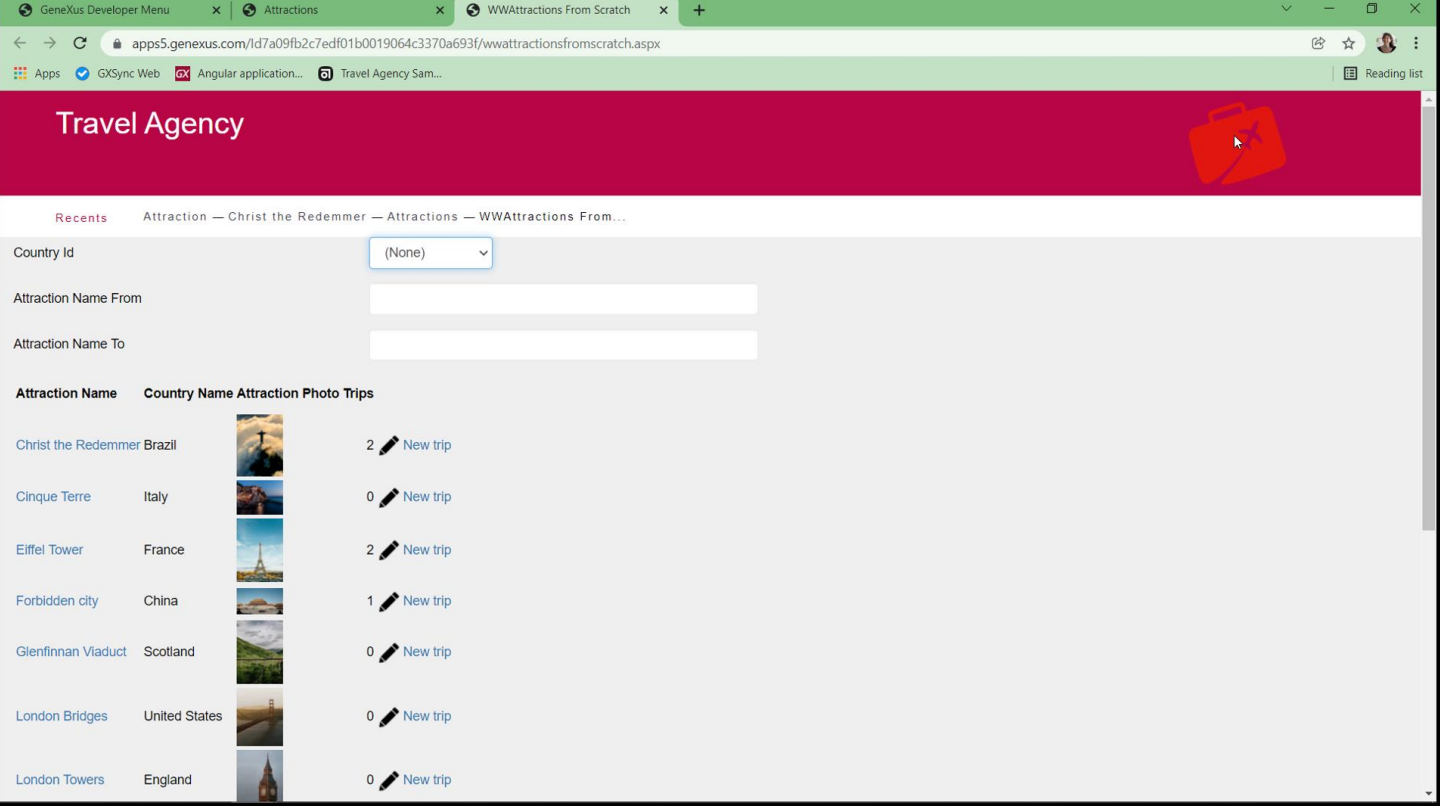


... in the transaction buttons or...



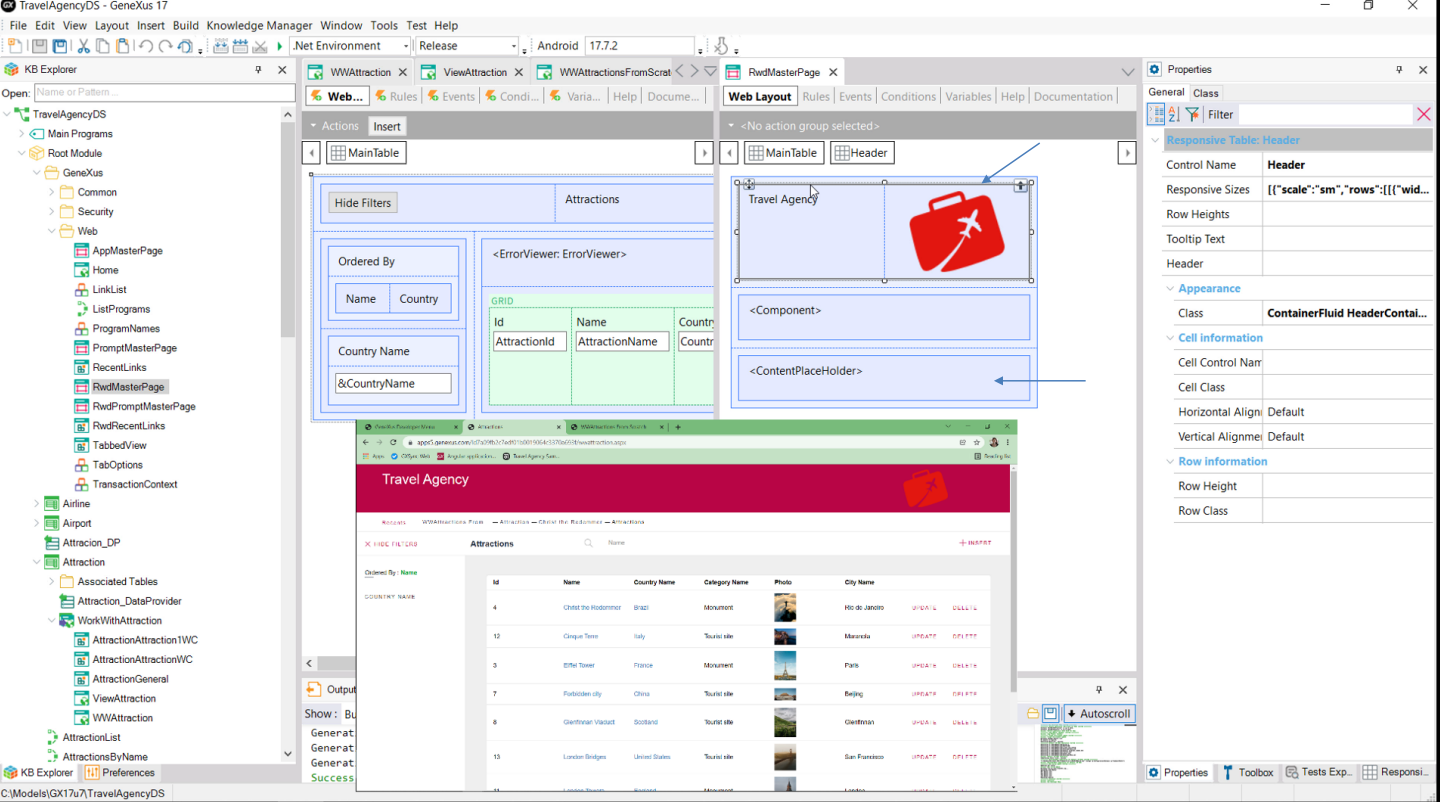
... those of the Web panel to view the information of an element.

There is also a common header that all pages share by default. Not only those of Work With...

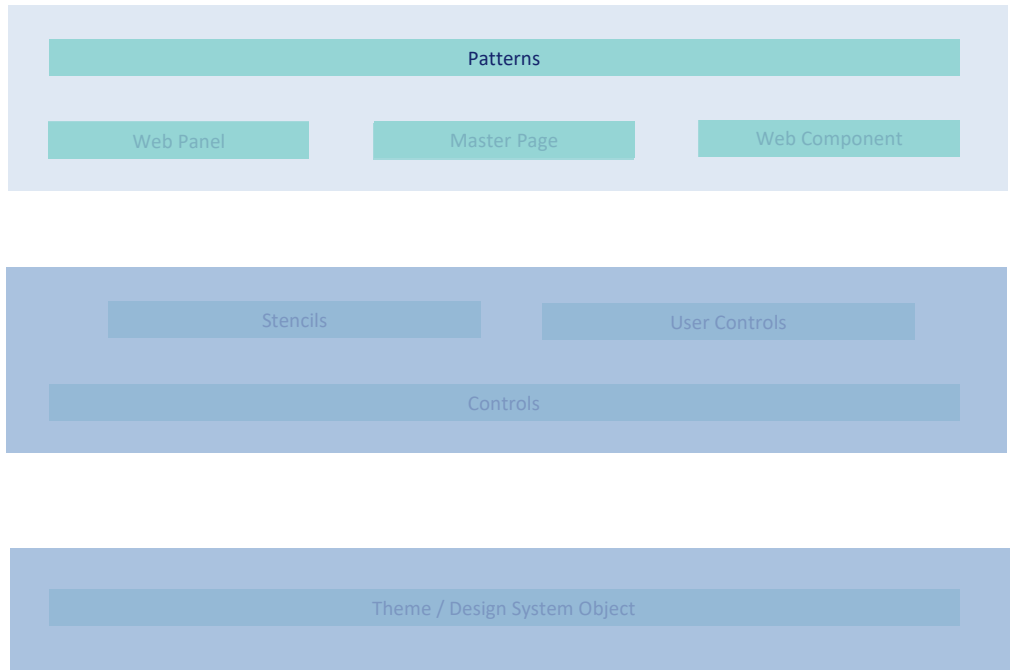


... but also those of the Web Panels we created from scratch.

To avoid repeating this common header in all Web Panels, there is a Master Page object.

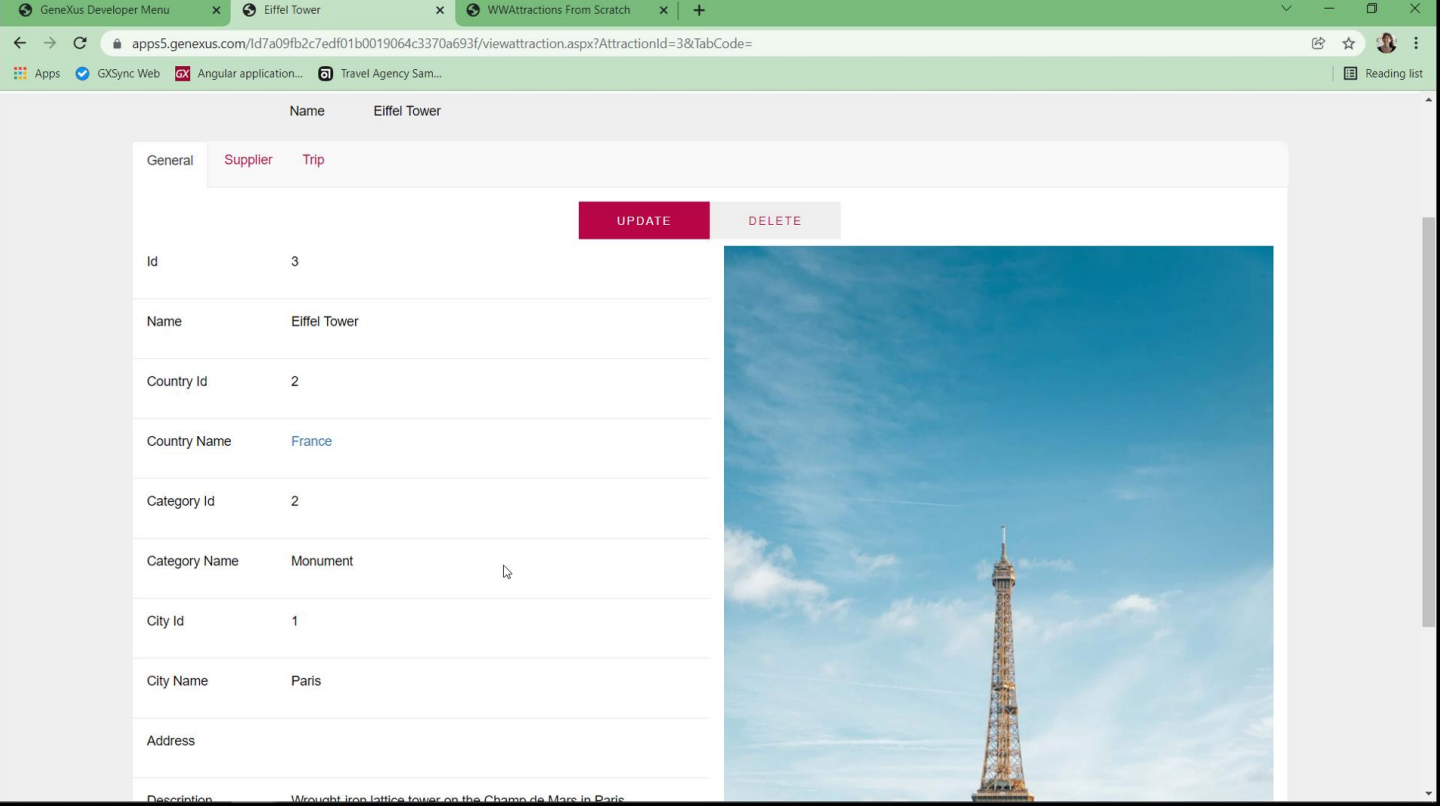


Each web panel will be loaded inside the ContentPlaceholder control of its Master Page. And here is that header we were looking at. When the Web Panel is executed, its Master Page is also executed and the page viewed in the Browser is the combination of both layouts, according to what the Master Page indicates.

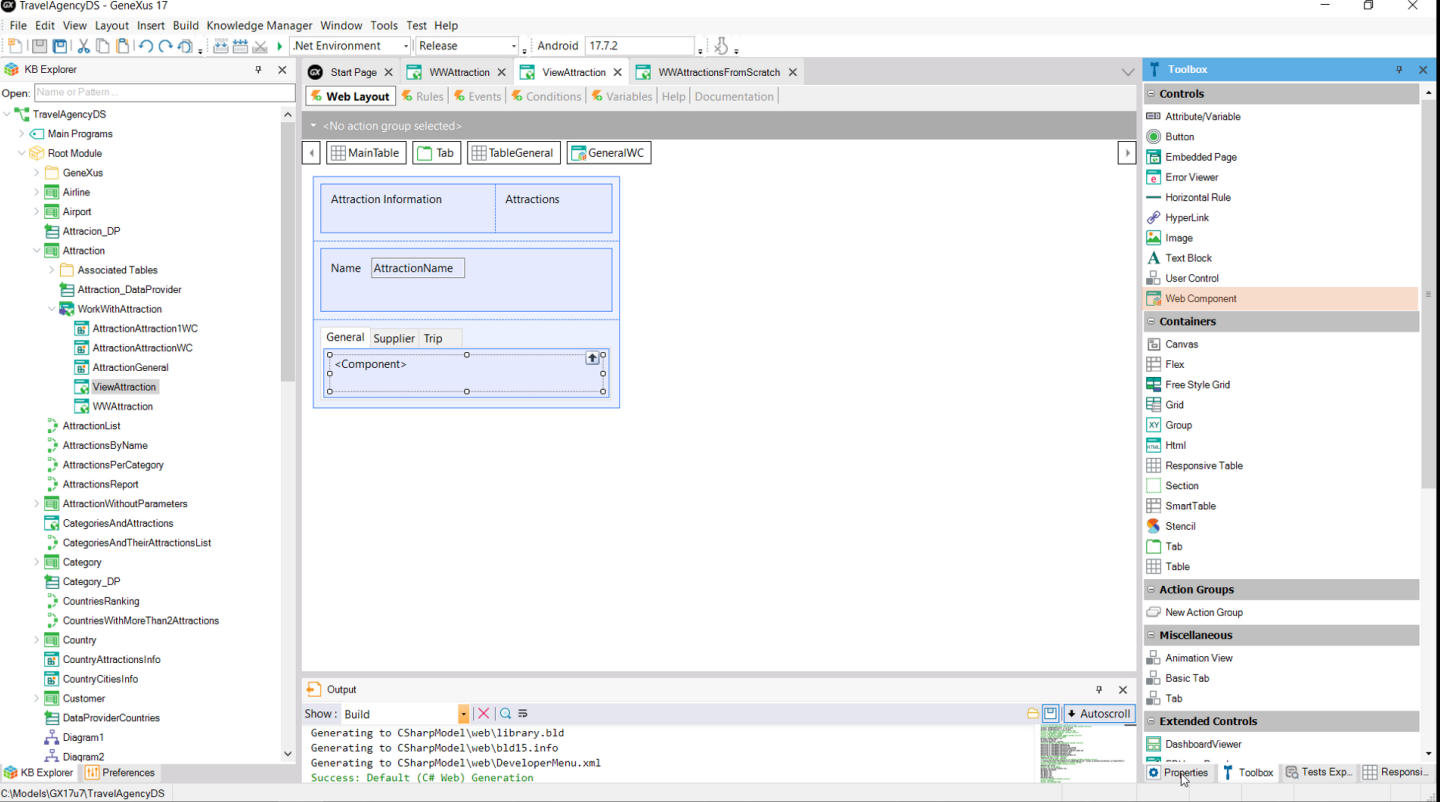
**DESIGN
SYSTEM**

But there is also another way to compose panels. They are Web Component objects; that is, they are like small pieces of Web panels that can be inserted into other Web Panels.

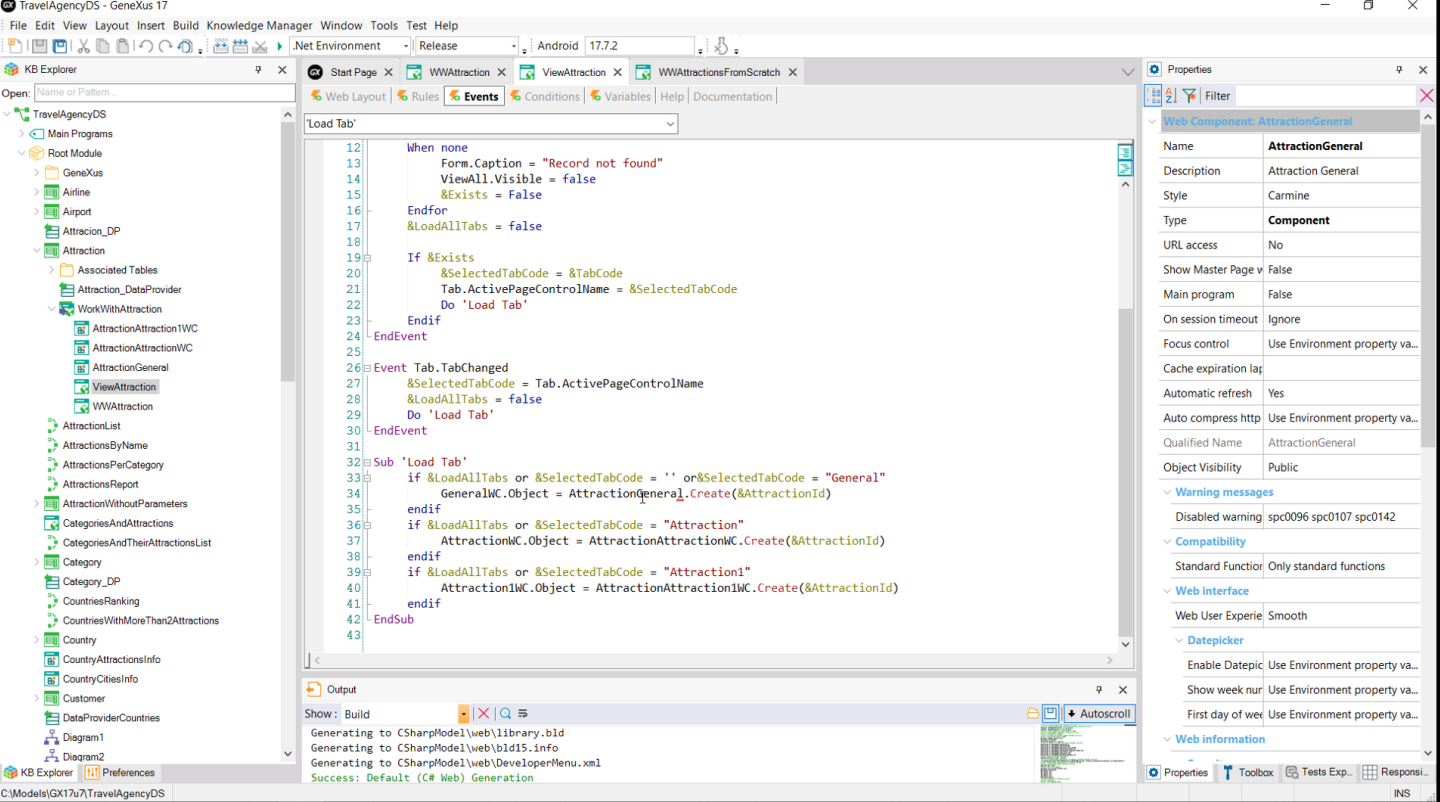
For example, the Work With pattern already did it.



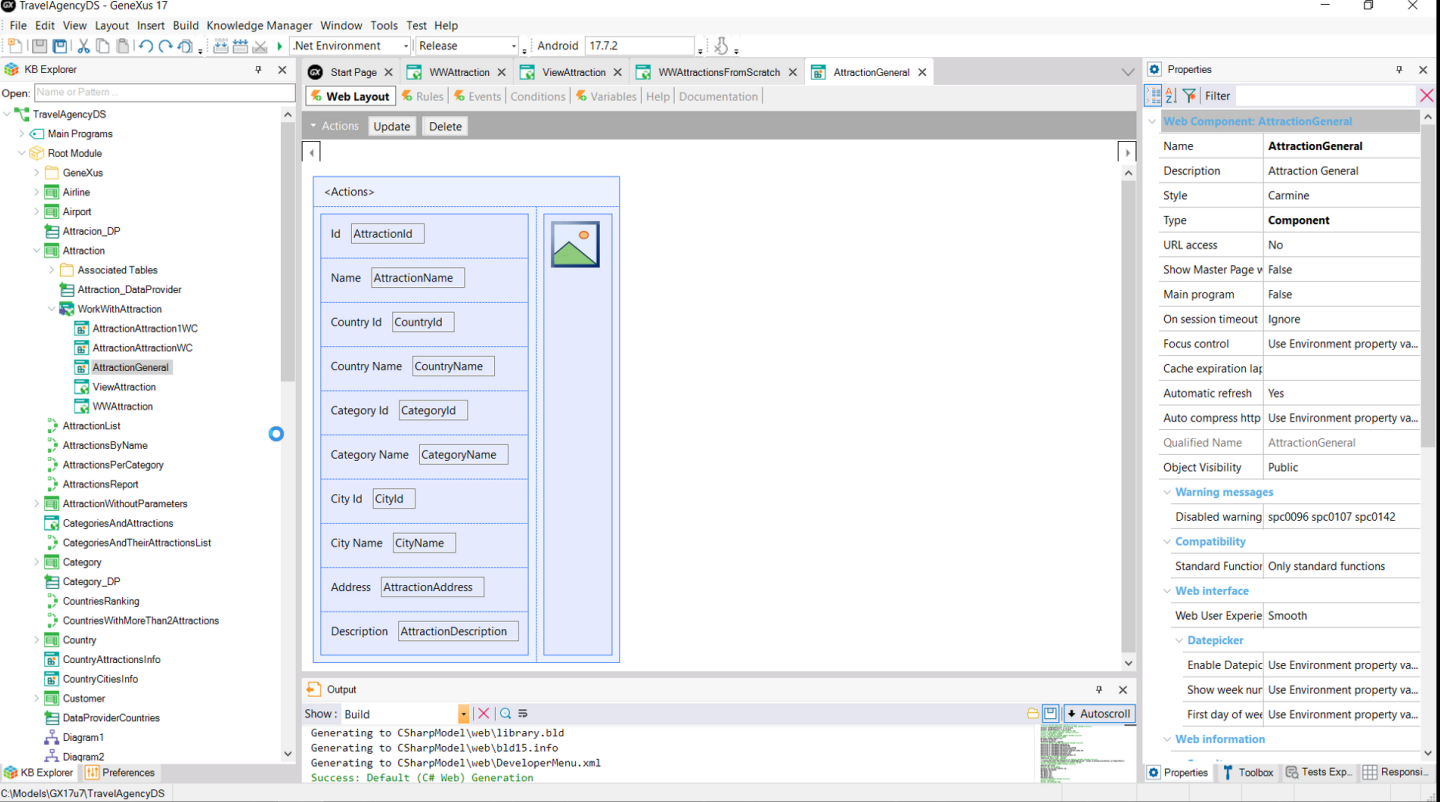
In the Web Panel showing a tourist attraction, we see three tabs, the first one showing the general information of the attraction.



In GeneXus, for the General tab we don't have attribute controls, but a control of component type.

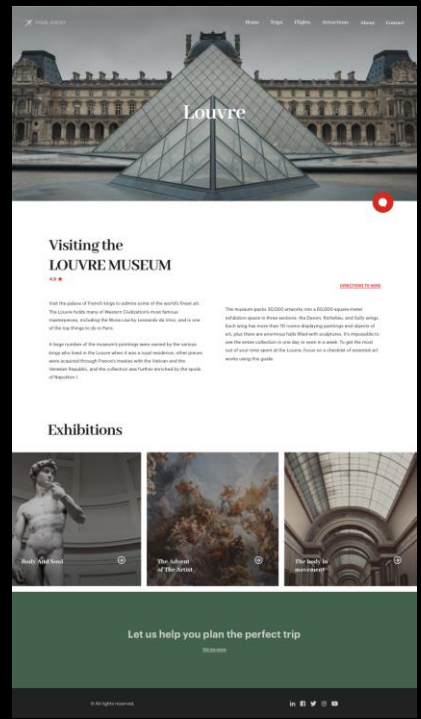
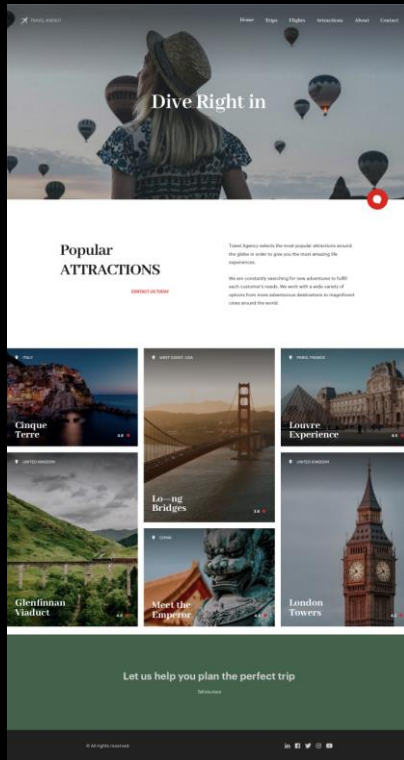
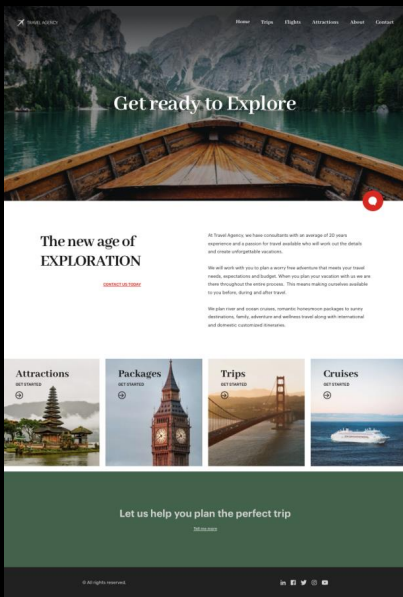


And in the events it is indicated that a Web Component should be loaded in there. The one named AttractionGeneral. We see that it is of Component type.



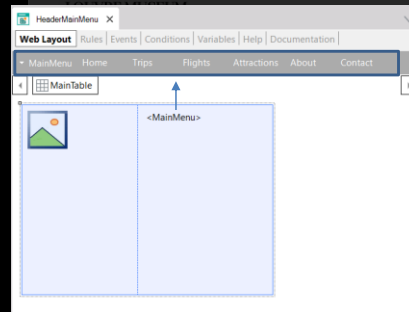
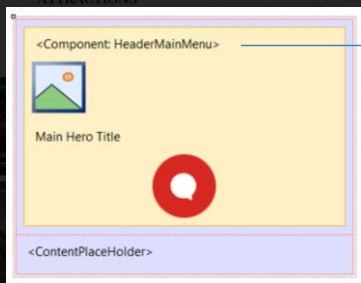
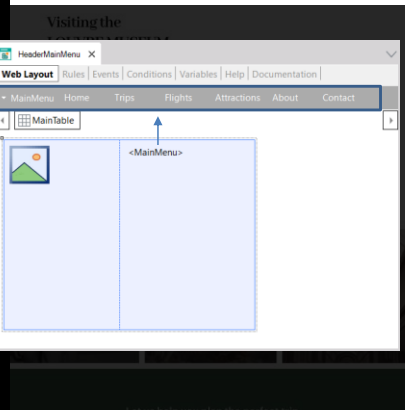
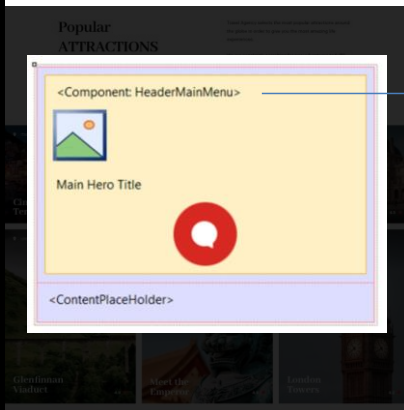
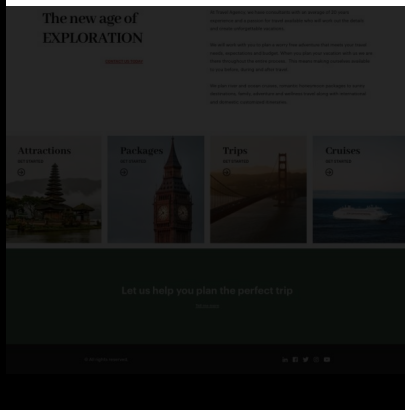
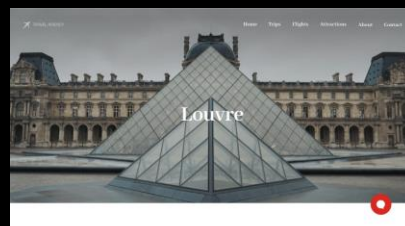
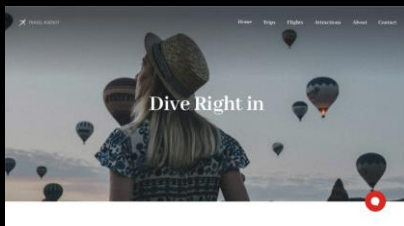
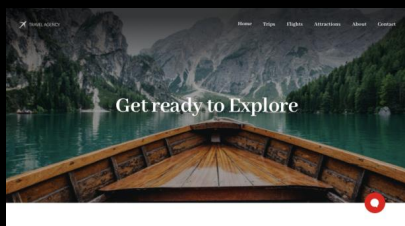
If we open it, it looks like an ordinary web panel, with its parm rule and events.

The pattern has already built its screens with these three objects, in order to reuse as much as possible. We should also use them to create high-quality, cost-effective systems that can be implemented once and reused as many times as necessary.



For example, suppose that now we are not interested in the back-office application that uses the pattern, but in the customer-facing one, i.e. the one that will be used by the end customers.

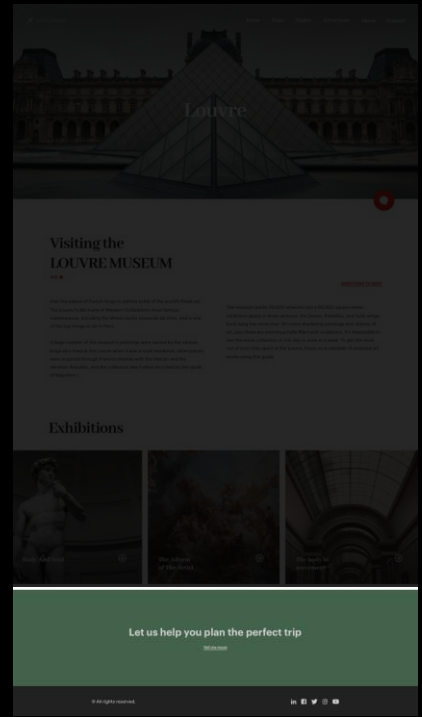
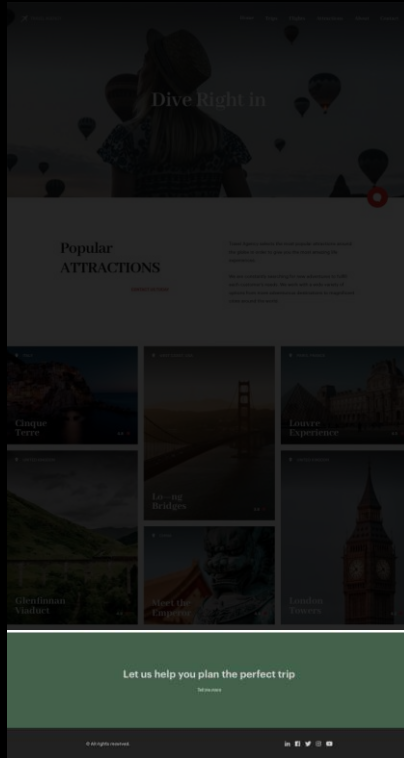
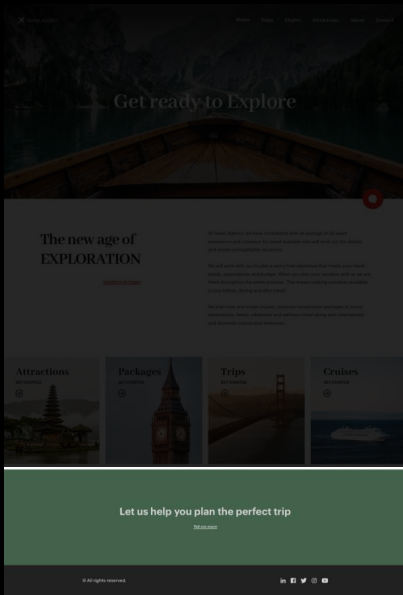
Suppose that the designers of our team designed the first three screens of the application. The Home, from which you call one that shows all the tourist attractions that can be visited; also, from this one, by choosing an attraction, the last one is called, which shows the information of that chosen tourist attraction.



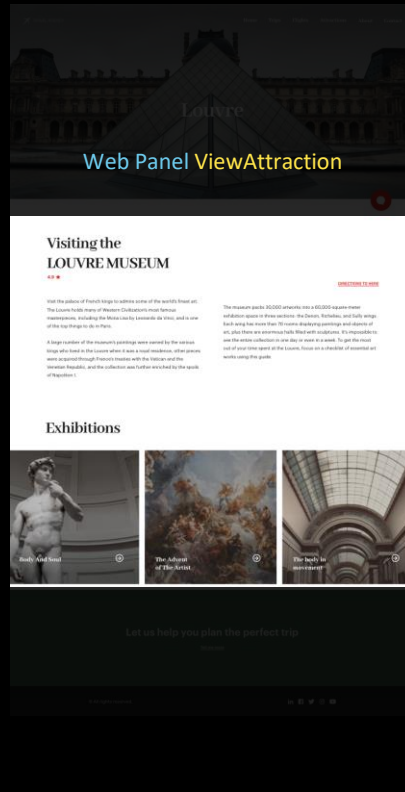
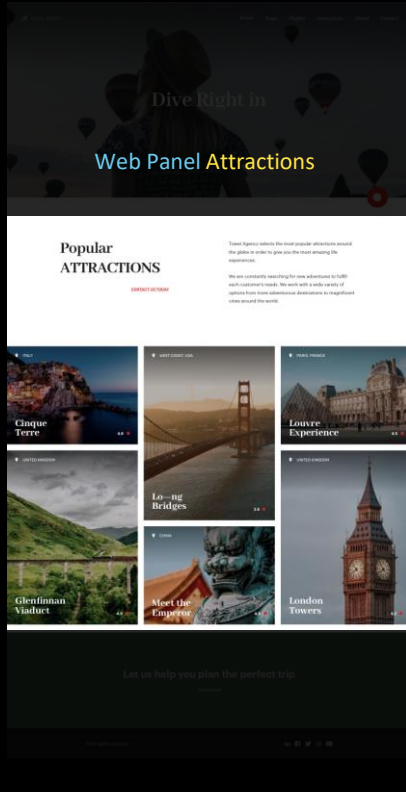
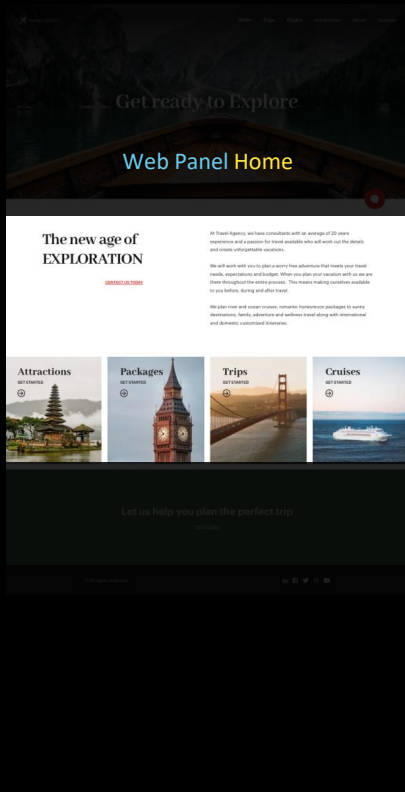
Note that from a design point of view there are repetitions: the three screens have a header with a background image, an overlay text, a menu and a logo, and an image representing a chatbot. Ideally, a Master Page should be used to implement this common header, where, depending on the object being loaded in the content container, the background image and text will be displayed (this is programmed in the Start event of the Master Page).

If we need to reuse the logo and menu somewhere else, we can choose not to implement them directly in the Master Page itself, but in a Web Component, which we insert in the Master Page... and everywhere else we need it, of course.

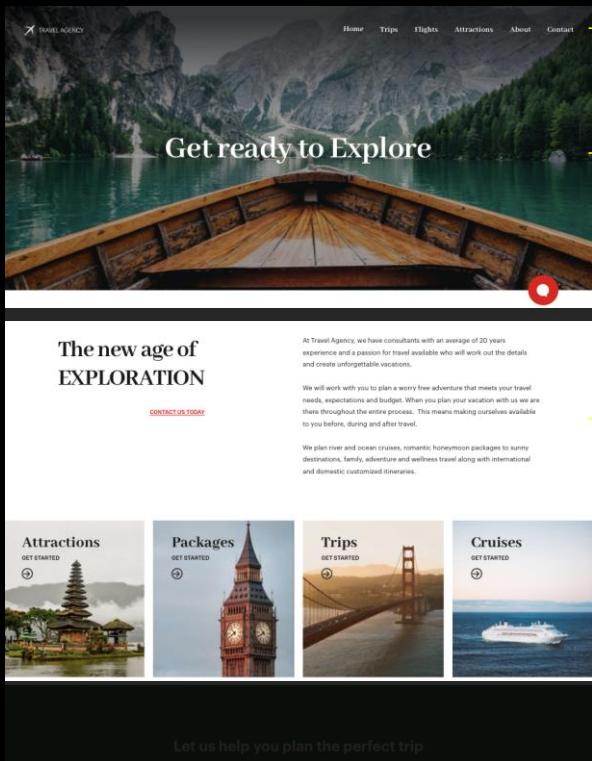
The web component has behavior: for example, we must program which object to call for each menu action. We also want to program an action when the user clicks on the logo. That's why it is not an object that allows reusing only one part of the screen, but rather it can be considered as a miniature Web Panel. For most purposes it is a Web Panel.



OK, let's go on... We didn't say it, but, of course, the common footer should also be placed in the Mater Page.



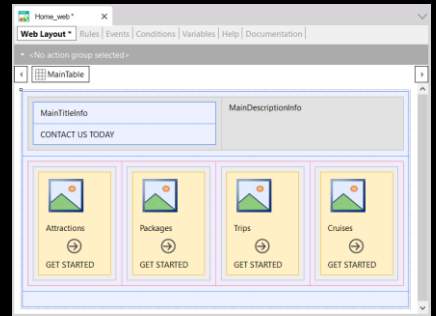
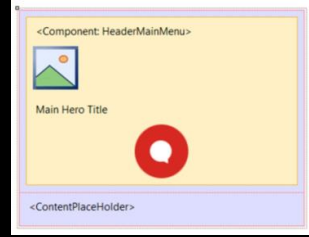
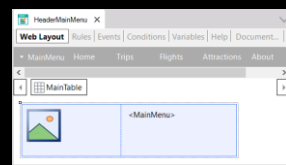
The specific content of each page will be implemented in each individual Web Panel.



→ Web Component

→ Master Page

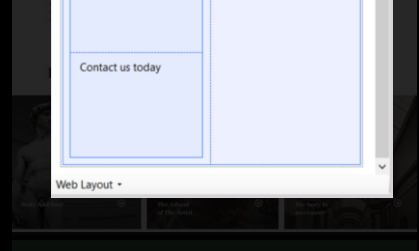
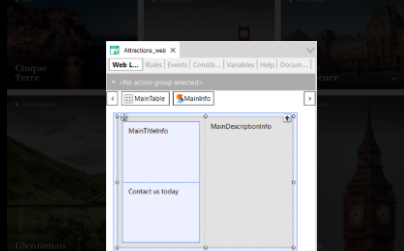
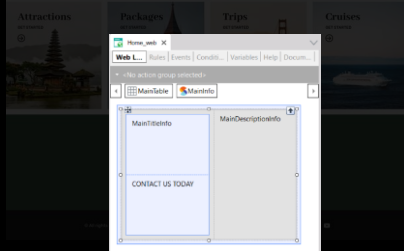
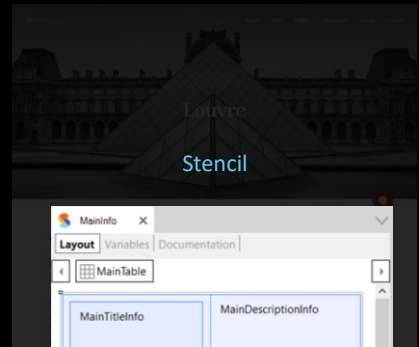
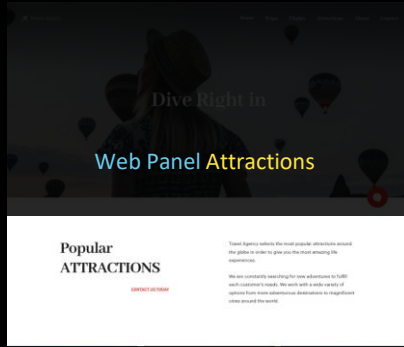
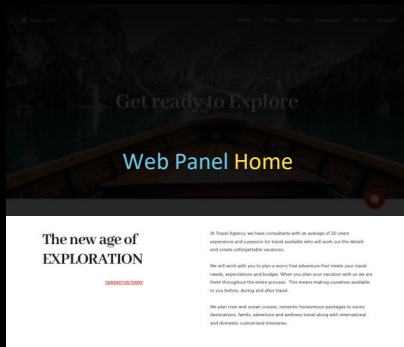
→ Web Panel



These objects (Web Panel, Master Page, and Web Component) are executable; that is, they have behavior.

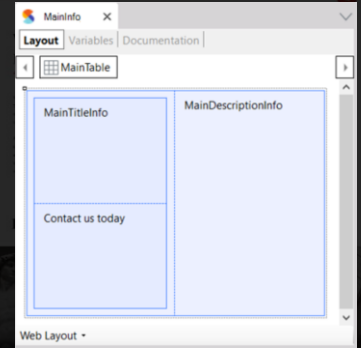
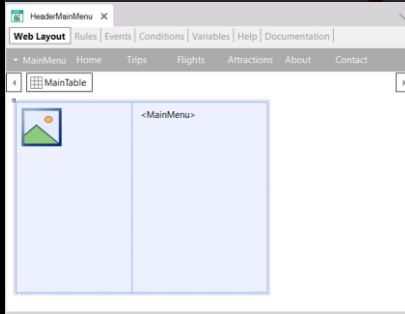
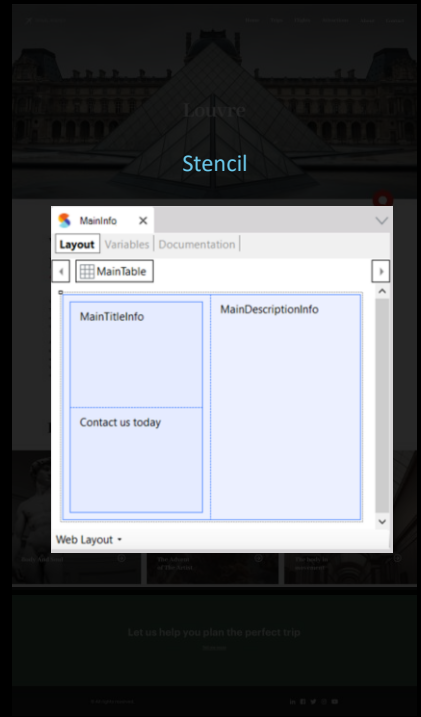
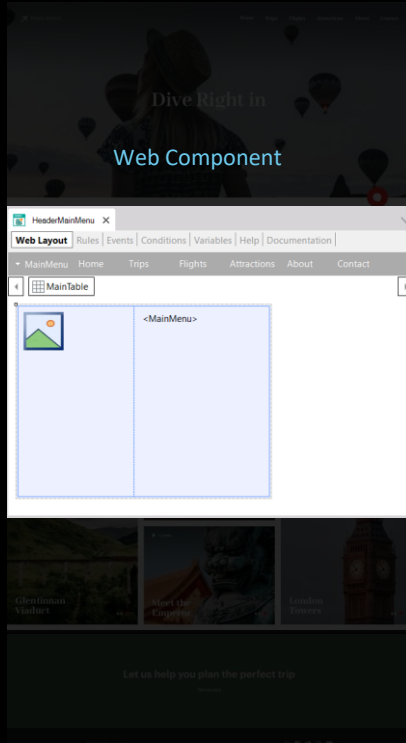
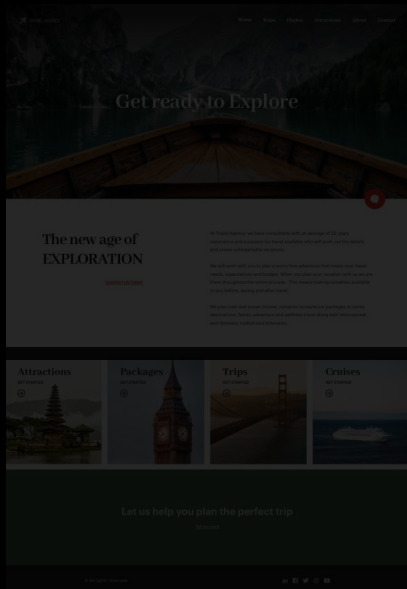
But there is also another type of object that does offer to reuse only a piece of layout, without behavior. It is the Stencil object, which allows doing the same thing that Web Components do, but only at screen level, so it will not have rules or events.

It is a way to build more complex controls from simpler controls.



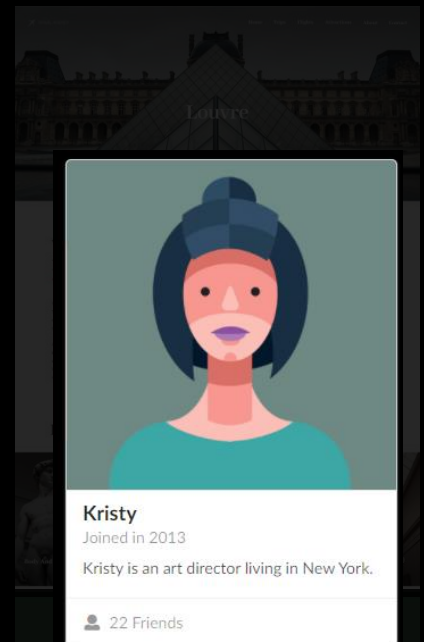
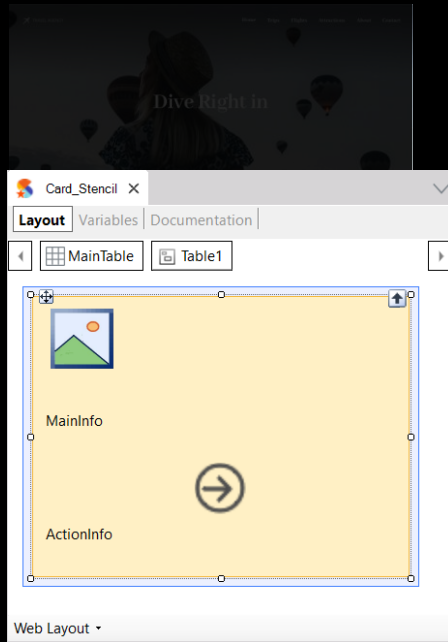
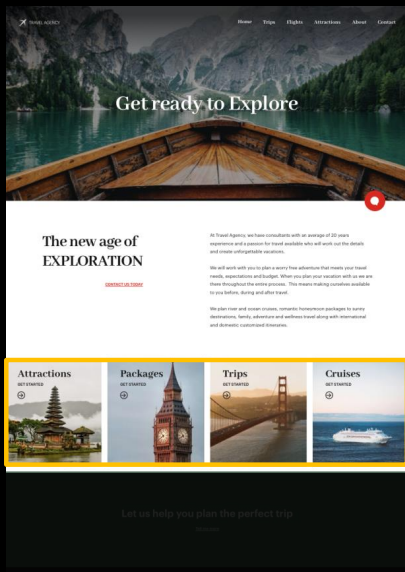
If we look at the first two Web Panels, we see that this screen section has the same structure and design. In both Web panels, we would have to repeat tables with text blocks and give them exactly the same layout. To avoid these repetitions, we can group these controls in a Stencil object, and then insert a stencil control in both Web Panels, which is loaded with the created stencil object.

In general, if we know that the structure and design of a set of controls on a screen will be repeated on many screens, even if they have different content, then we can group them in a Stencil object and use that object in the Web Panel, Master Page, or Web Component.



The main difference between a Web Component and a Stencil is that the Stencil has no behavior; that is, it is not an executable object, while the Web Component is.

This is why the Stencil is clearly a Design object. It is entirely intended for the Design System.



Another example of use of Stencils: we want to implement the menu with images that we see on the home page. There are 4 options with identical design. They are like cards. Then we could create a Stencil and reuse it 4 times.

In this Stencil, we would have to overlay two texts and an arrow image on a background image in order to create this kind of card. That is, we are composing a kind of larger control using tables, images, variables, textblocks. And the question that we will answer in a moment is how we design all these controls.

The other option, without using a stencil, is to directly incorporate a control made by a third party. For example, the control to design cards provided by the SemanticUI design framework.

Card ×

Screen Template Properties Documentation

```
1 <div class="ui card">
2   <div class="image">
3     
6     <a class="header">{{MainInfo}}</a>
7     <div class="meta">
8       <span class="date">{{SecondaryInfo}}</span>
9     </div>
10    <div class="description">
11      {{DescriptionInfo}}
12    </div>
13  </div>
14  <div class="extra content" {{OnClick}}>
15    <a>
16      <i class="arrow alternate circle right">
17        {{ExtraInfo}}
18      </i>
19    </a>
20  </div>
```

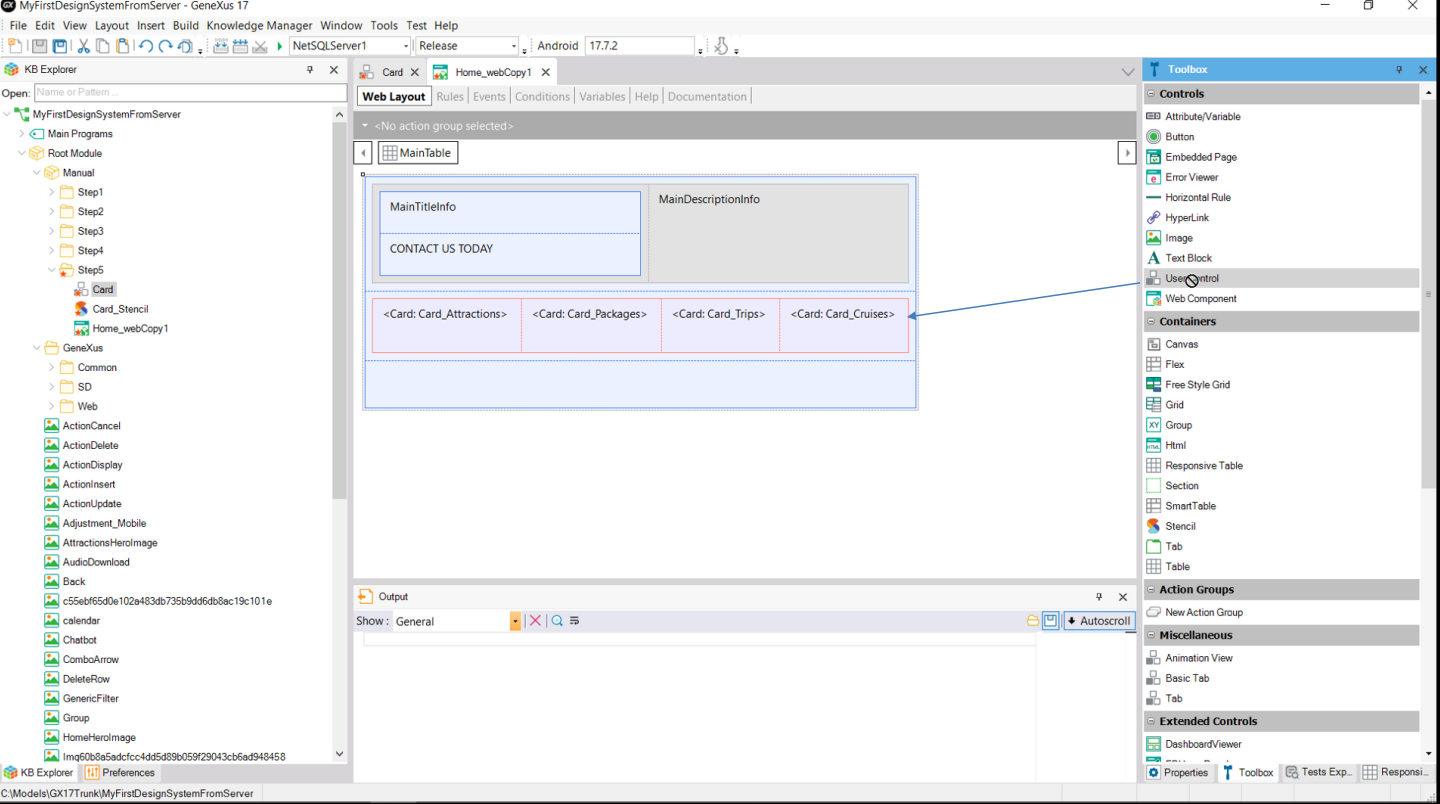
Properties

User Control: Card

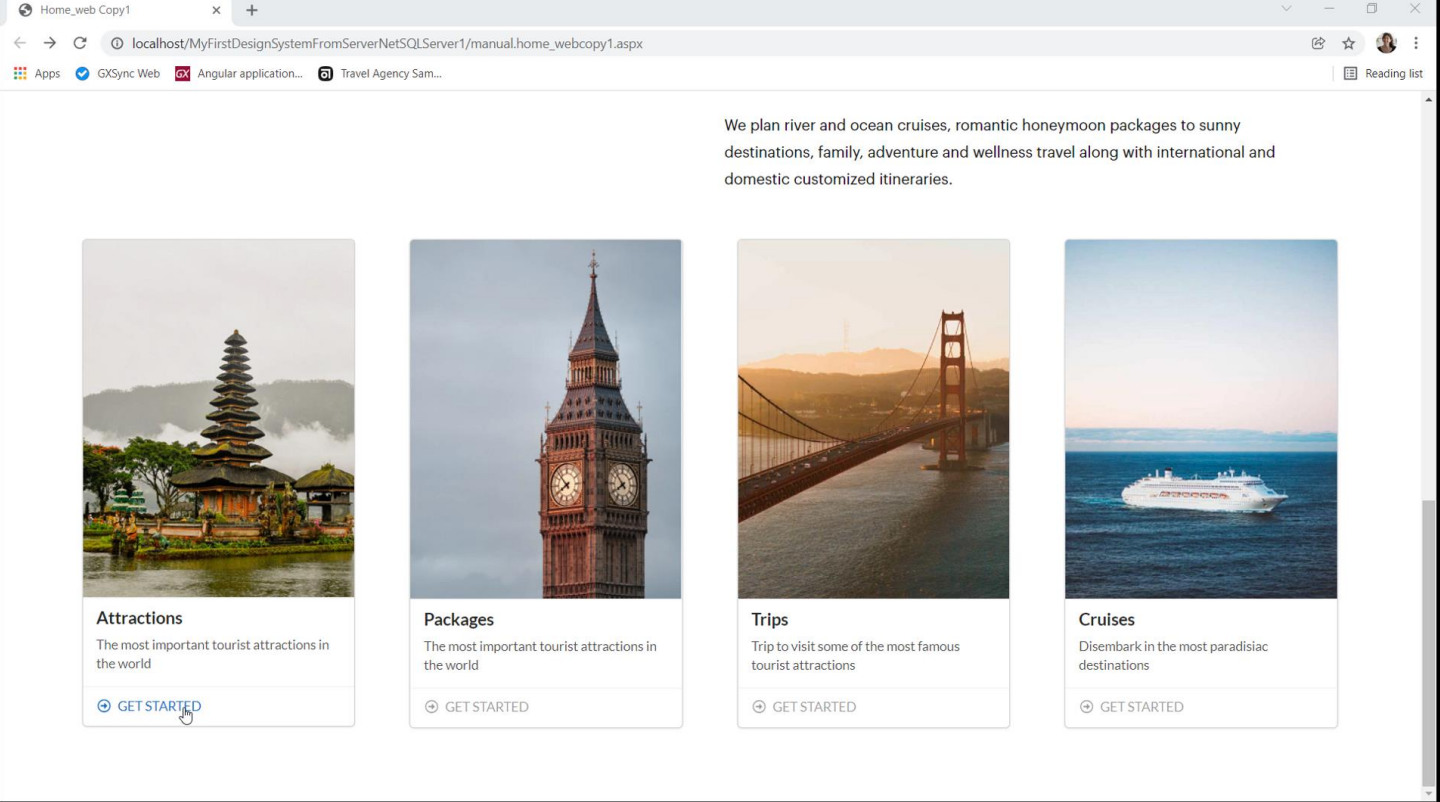
Name	Card
Description	Card
Module/Folder	Step5
Is Control Type	False
References	
Base Control Type	None
Base CSS	Semantic-UI-master-default
Qualified Name	Manual.Card
Object Visibility	Public



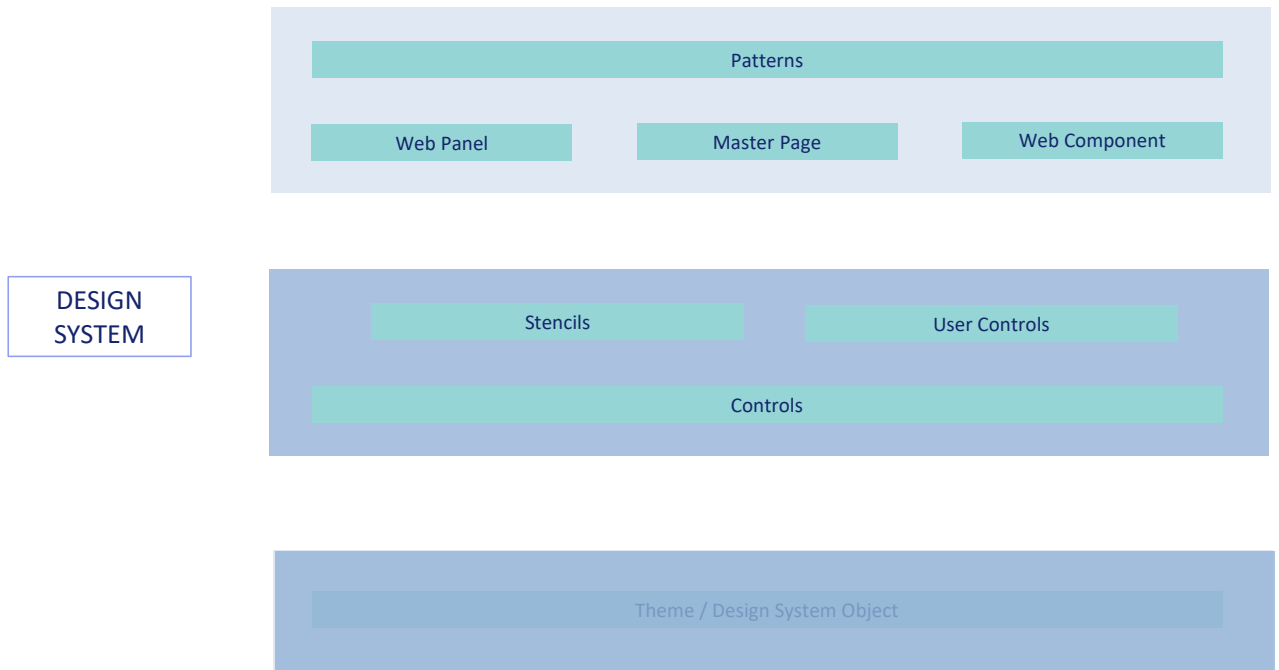
To do so, it is enough to create a User Control object in GeneXus. Then, copy the HTML code provided by the supplier, change the fixed data to variable data, and indicate the file containing the style of the HTML elements. That is, the CSS file provided by the supplier.



After doing this, we can use it as another control from the toolbox.



And so, at runtime we will see...



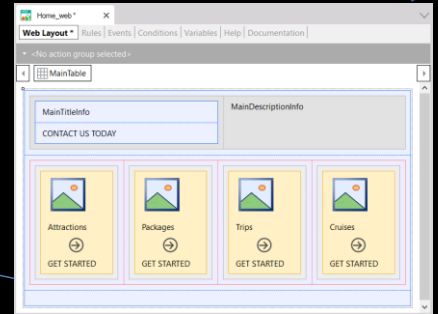
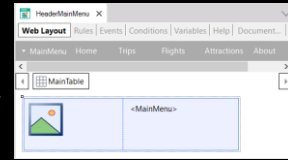
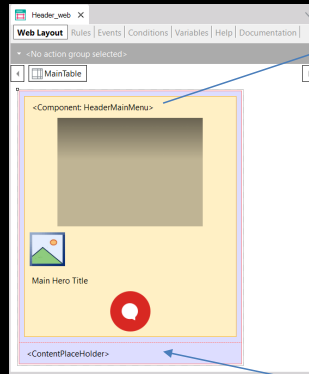
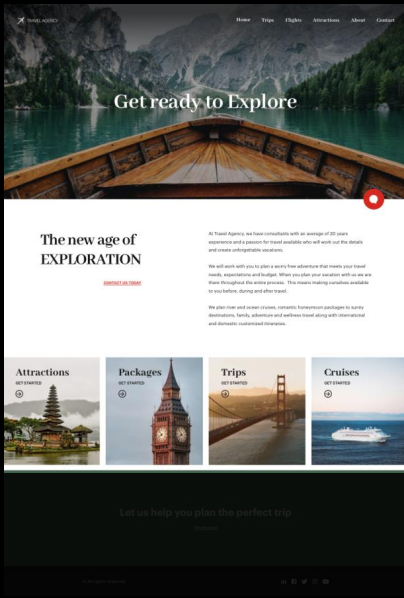
In short: first we talked about the patterns that use Web Panels, Master Page, Web Components to implement parts of the application.

We've seen these objects as ways to componentize the programs for maximum reuse. Also, that they are objects with a layout and behavior.

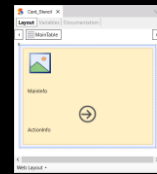
Now we are getting close to the heart of the matter. In any of these objects we have a layout to be designed.

And, besides being able to use third-party controls that already come with design—User Controls— in that layout we can use Stencils to compose controls. But adding design to a Stencil is the equivalent to adding design to any set of controls in a layout.

So now we're getting to where we wanted: regardless of where the GeneXus controls are located, how do we add design features to them?

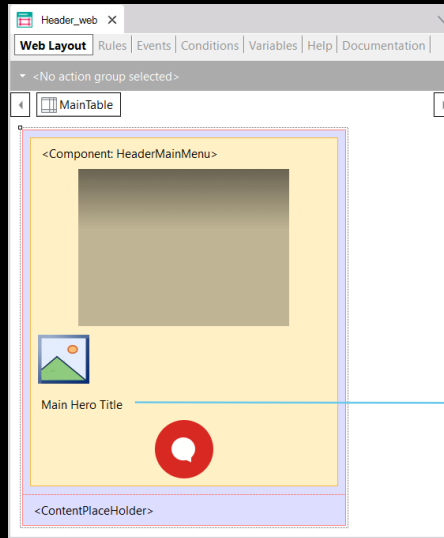
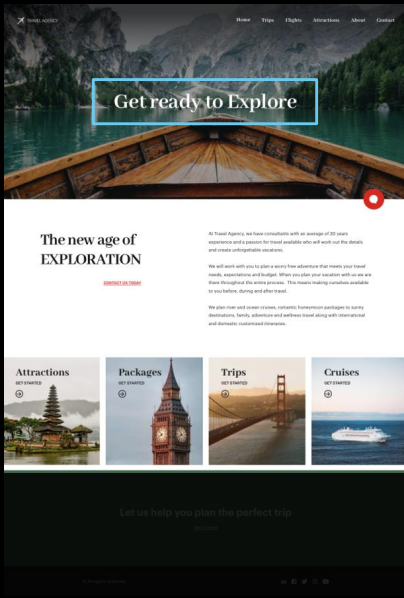


control → class ...



On the left is the screen as we want to see it at runtime. On the right is the Web Panel Home. It is loaded inside the displayed Master Page, which in turn uses the Web Component. In addition, besides having common controls, the Web Panel uses a Stencil that also has a layout.

The way to add design to each one of the controls of these layouts is using what we know as **classes**. Each control is associated with one or more classes, which determine how it is displayed on the screen.



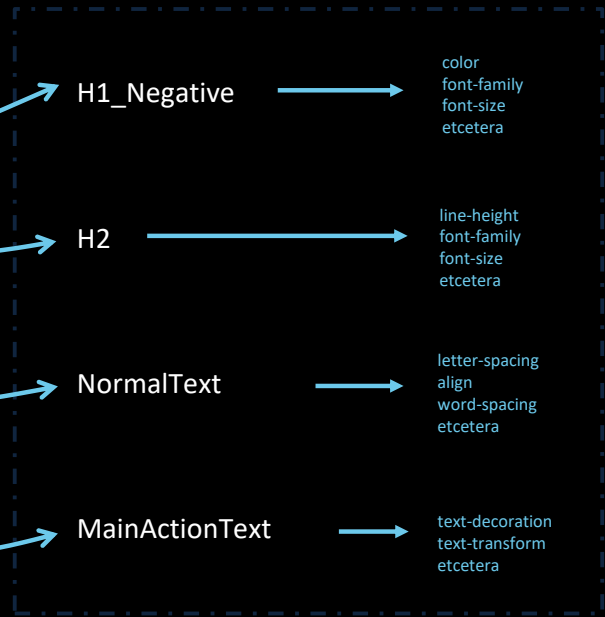
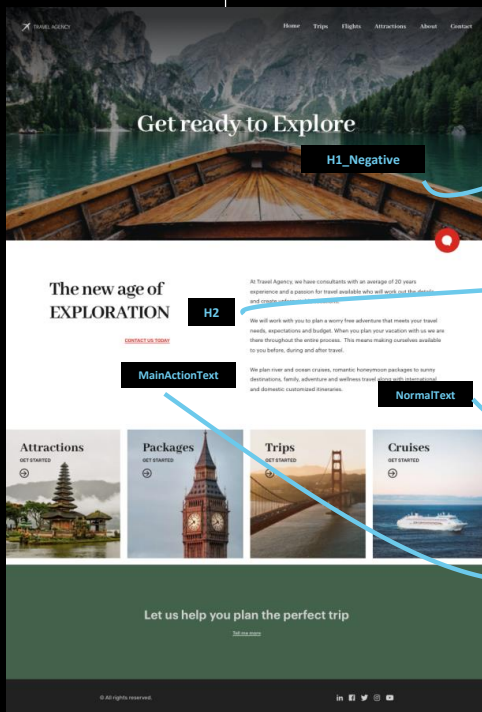
H1_Negative

color: white;
font-family: AbhayaLibre-Bold;
font-size: 95px;

For example, if we look at the title “Get ready to Explore” corresponding to the text block we see here... how do we set it to be a white title, to have that font family, and that size?

The answer is: by associating it with a class, which is named as we want and is semantically meaningful, for example, H1_Negative, to indicate headings on a dark background. Also, indicating the desired design features at the class level.

The class can be reused in many other controls, and that is the advantage of its design features being independent of the control.

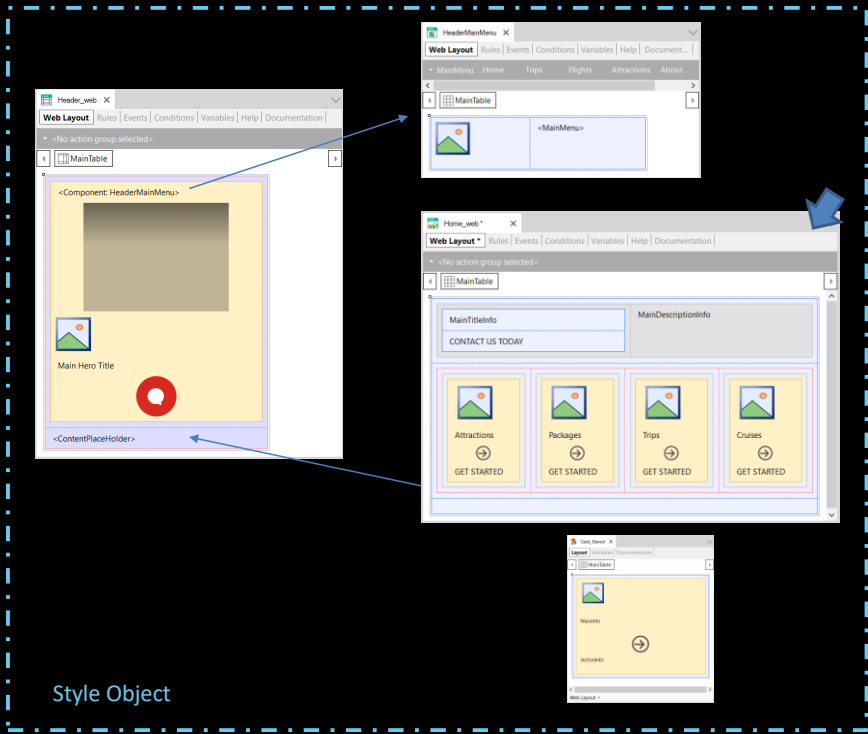
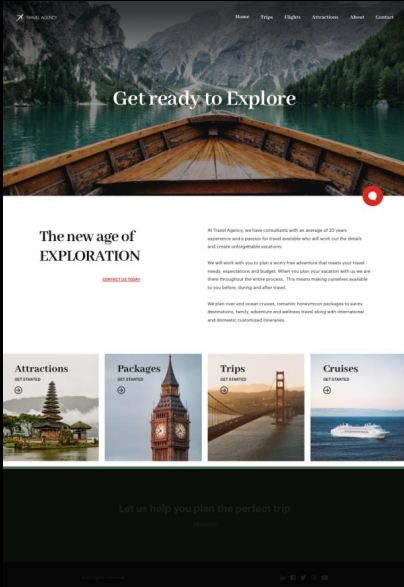


So, the idea is to associate all UI controls or elements in the objects—for example, these 4 texts—with the style classes identified as significant. Then it will only be necessary to define the design properties of each of these classes, in an object that centralizes them.

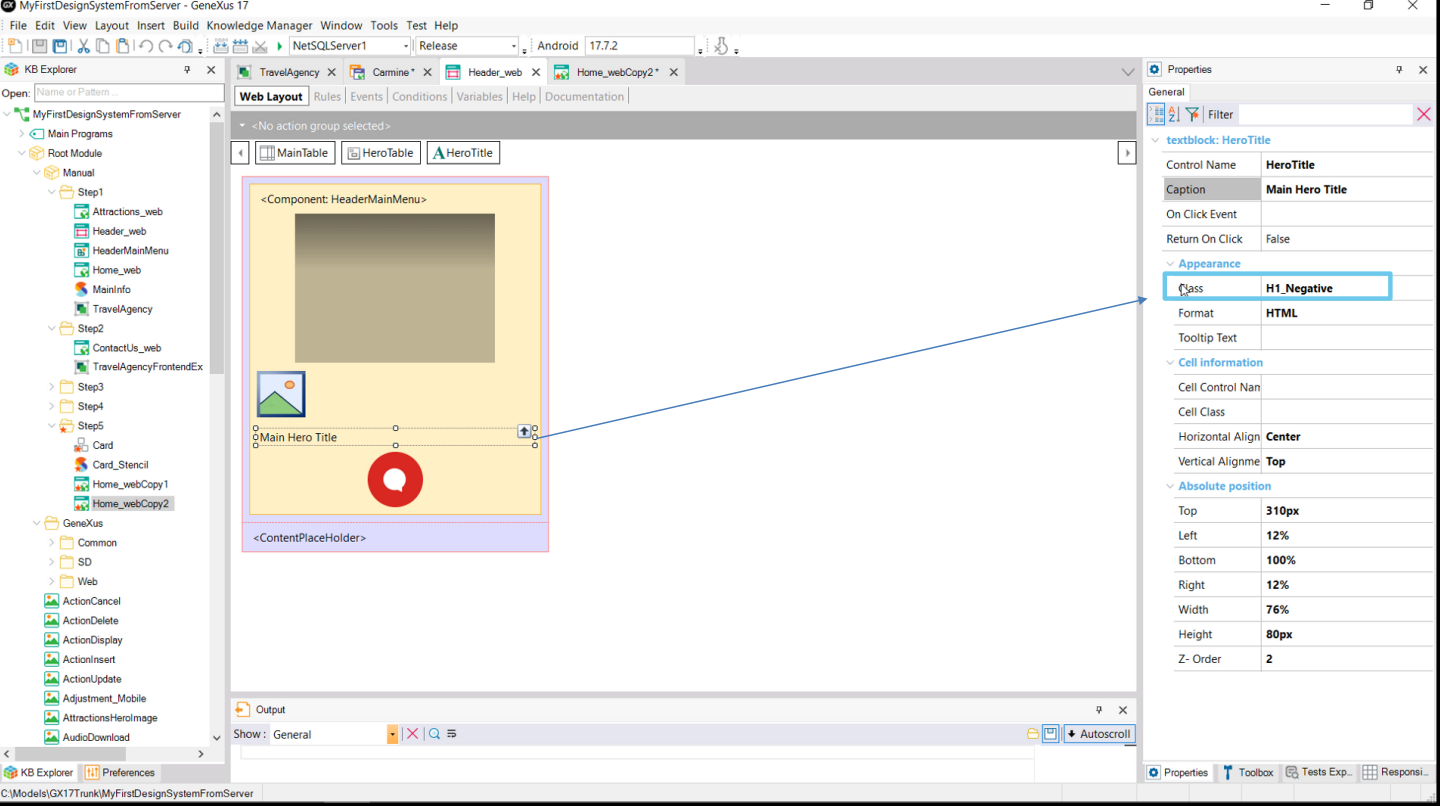
Until GeneXus 17, this object was the Theme, but from then on it can also be its evolution, the Design System object.

Therefore, the page, which has class names for each control, will only need to know which Design System object or Theme it has to go to in order to retrieve the design properties of each one of those classes.

And that is the way GeneXus separates the structure and content of the pages from their design.



All these objects that implement the page we will see in the browser have an associated style object: either the Theme object or the new Design System object. In this Theme or Design System object, the classes are declared with their specific properties that give them the style they will be displayed with on the screen.



MyFirstDesignSystemFromServer - GeneXus 17

File Edit View Layout Insert Build Knowledge Manager Window Tools Test Help

NetSQLServer1 Release Android 17.7.2

KB Explorer

TravelAgency x Carmine * x Header_web x Home_webCopy2* x

Web Layout Rules Events Conditions Variables Help Documentation

<No action group selected>

MainTable MainInfo MainInfo_Table1 MainInfo_MainTitleInfo

MainTitleInfo MainDescriptionInfo

CONTACT US TODAY

Attractions Packages Trips Cruises

GET STARTED GET STARTED GET STARTED GET STARTED

Output

Show: General

Properties

General

Filter

textblock: MainInfo_MainTitleInfo

Caption	MainTitleInfo
On Click Event	
Return On Click	False

Appearance

Class	H2
-------	----

KB Explorer Preferences

C:\Models\GX17Trunk\MyFirstDesignSystemFromServer

File Edit View Layout Insert Build Knowledge Manager Window Tools Test Help

NetSQLServer1 | Release | Android 17.7.2

KB Explorer | TravelAgency | Carmine | Header_web | Home_webCopy2

Open: Name or Pattern

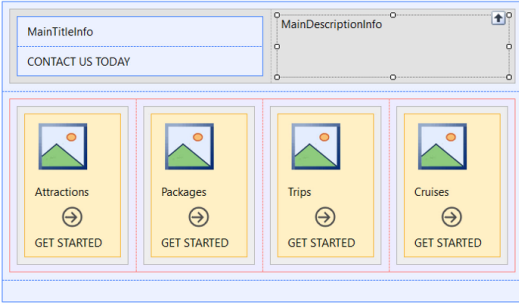
MyFirstDesignSystemFromServer

- Main Programs
- Root Module
- Manual
 - Step1
 - Attractions_web
 - Header_web
 - HeaderMainMenu
 - Home_web
 - MainInfo
 - TravelAgency
 - Step2
 - ContactUs_web
 - TravelAgencyFrontendEx
 - Step3
 - Step4
 - Step5
 - Card
 - Card_Stencil
 - Home_webCopy1
 - Home_webCopy2
- GeneXus
 - Common
 - SD
 - Web
 - ActionCancel
 - ActionDelete
 - ActionDisplay
 - ActionInsert
 - ActionUpdate
 - Adjustment_Mobile
 - AttractionsHeroImage
 - AudioDownload

Web Layout | Rules | Events | Conditions | Variables | Help | Documentation

<No action group selected>

MainTable | MainInfo | MainInfo_MainDescriptionInfo



Output

Show: General | Autoscroll

Properties

Web Panel: Home_webCopy2

Name	Home_webCopy2
Description	Home_web_Copy2
Module/Folder	Steps5
Style	TravelAgency
Type	Web Page
Master Page	Header_web
Show Master Page v	False
Main program	True
On session timeout	Ignore
Focus control	Use Environment property va...
Cache expiration lag	
Automatic refresh	Yes
Auto compress http	Use Environment property va...
Qualified Name	Manual.Home_webCopy2
Object Visibility	Public

Main object properties

Web Application	Default
Location	
Generator	Default (C# Web)

Warning messages

Disabled messages: spc0096 spc0107 spc0142

Compatibility

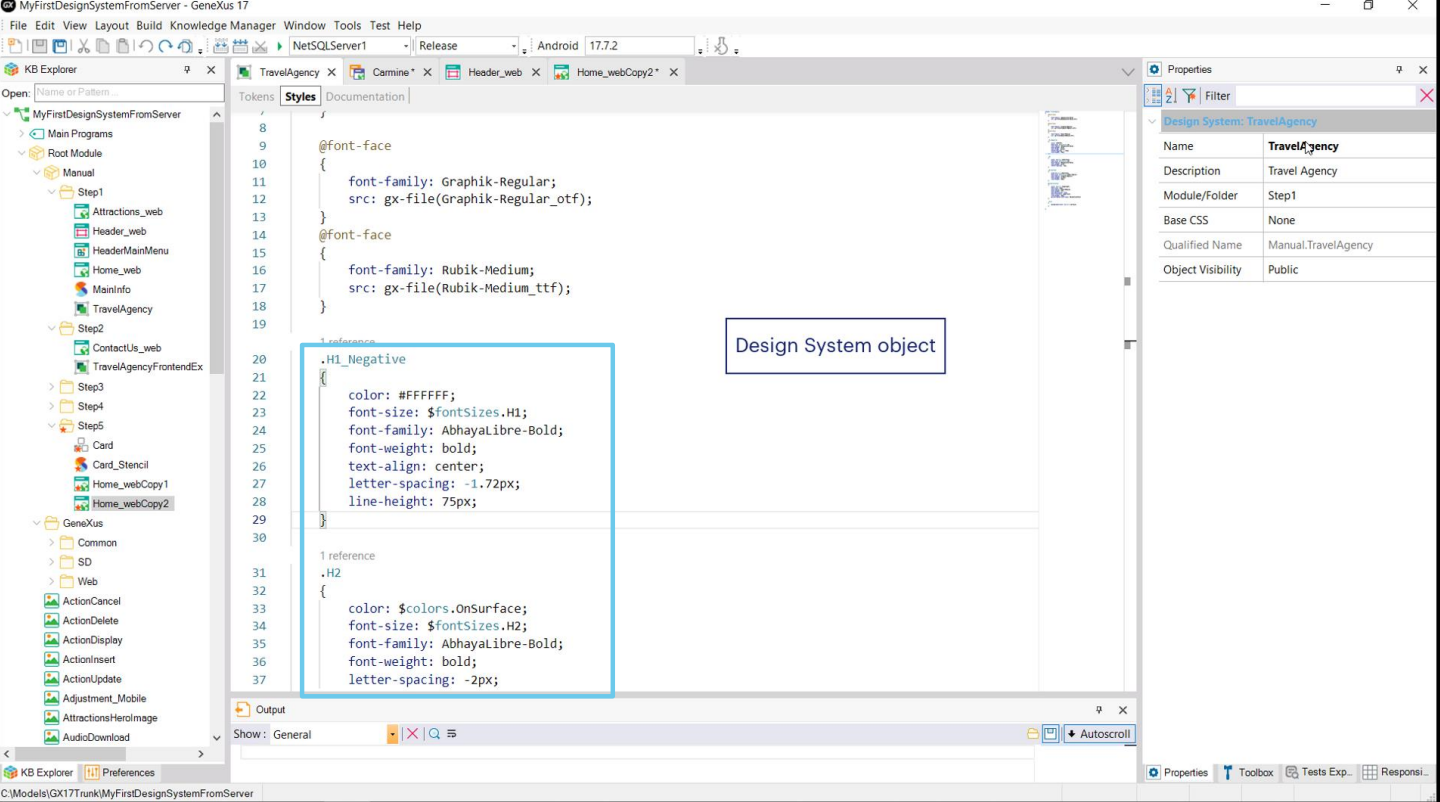
Standard Function: Only standard functions

Web interface

Web User Experie: Smooth

KB Explorer | Preferences

C:\Models\GX17Trunk\MyFirstDesignSystemFromServer



File Edit View Layout Insert Build Knowledge Manager Window Tools Test Help

NetSQLServer1 | Release | Android 17.7.2

KB Explorer | TravelAgency | Carmine | Header_web | Home_webCopy2

Web Layout | Rules | Events | Conditions | Variables | Help | Documentation

Open: Name or Pattern

MyFirstDesignSystemFromServer

- Main Programs
- Root Module
 - Manual
 - Step1
 - Attractions_web
 - Header_web
 - HeaderMainMenu
 - Home_web
 - MainInfo
 - TravelAgency
 - Step2
 - ContactUs_web
 - TravelAgencyFrontendEx
 - Step3
 - Step4
 - Step5
 - Card
 - Card_Stencil
 - Home_webCopy1
 - Home_webCopy2
 - GeneXus
 - Common
 - SD
 - Web
 - ActionCancel
 - ActionDelete
 - ActionDisplay
 - ActionInsert
 - ActionUpdate
 - Adjustment_Mobile
 - AttractionsHeroImage
 - AudioDownload

Web Layout

<No action group selected>

MainTable | MainInfo | MainInfo_MainDescriptionInfo

MainTitleInfo | MainDescriptionInfo

CONTACT US TODAY

Attractions | Packages | Trips | Cruises

GET STARTED | GET STARTED | GET STARTED | GET STARTED

Output

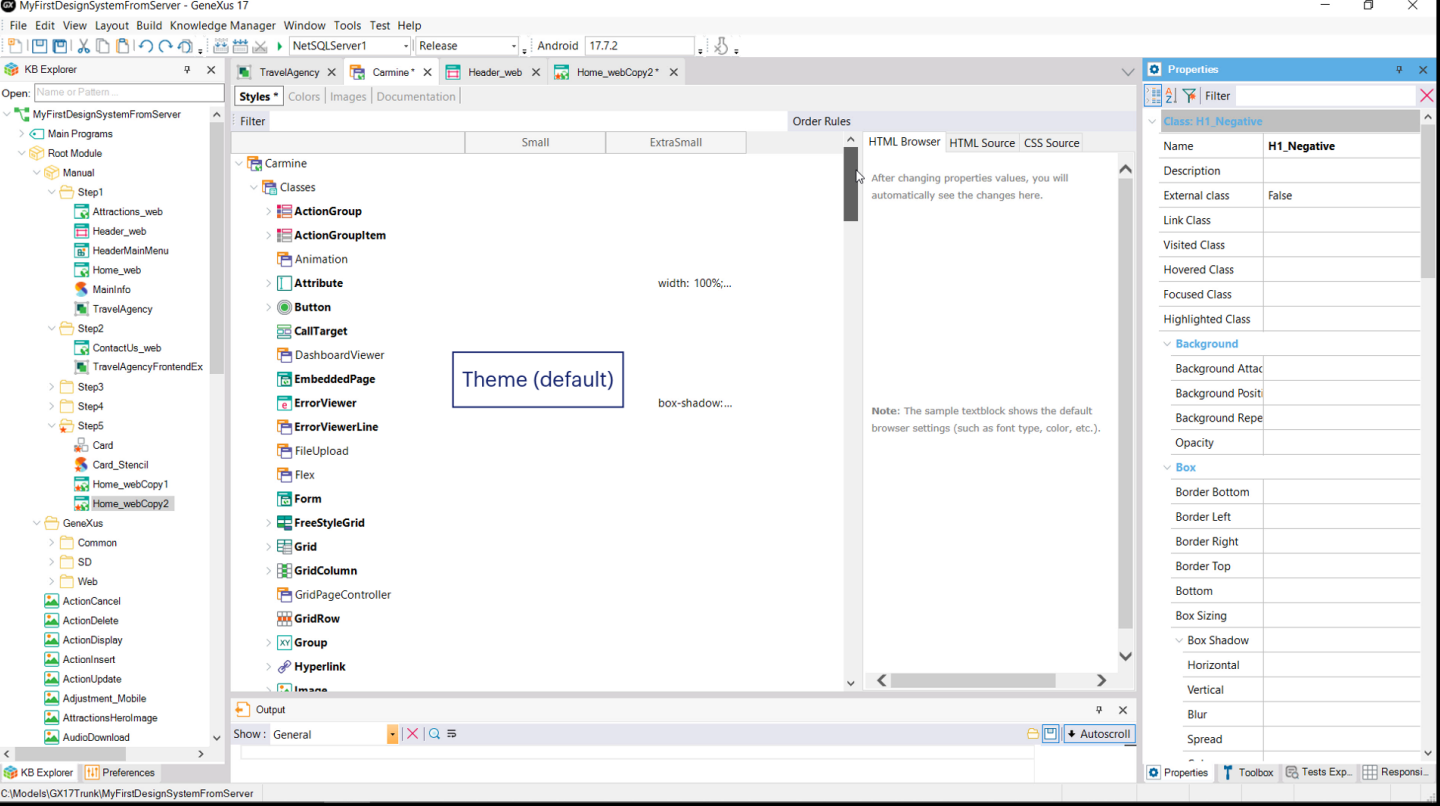
Show: General | Autoscroll

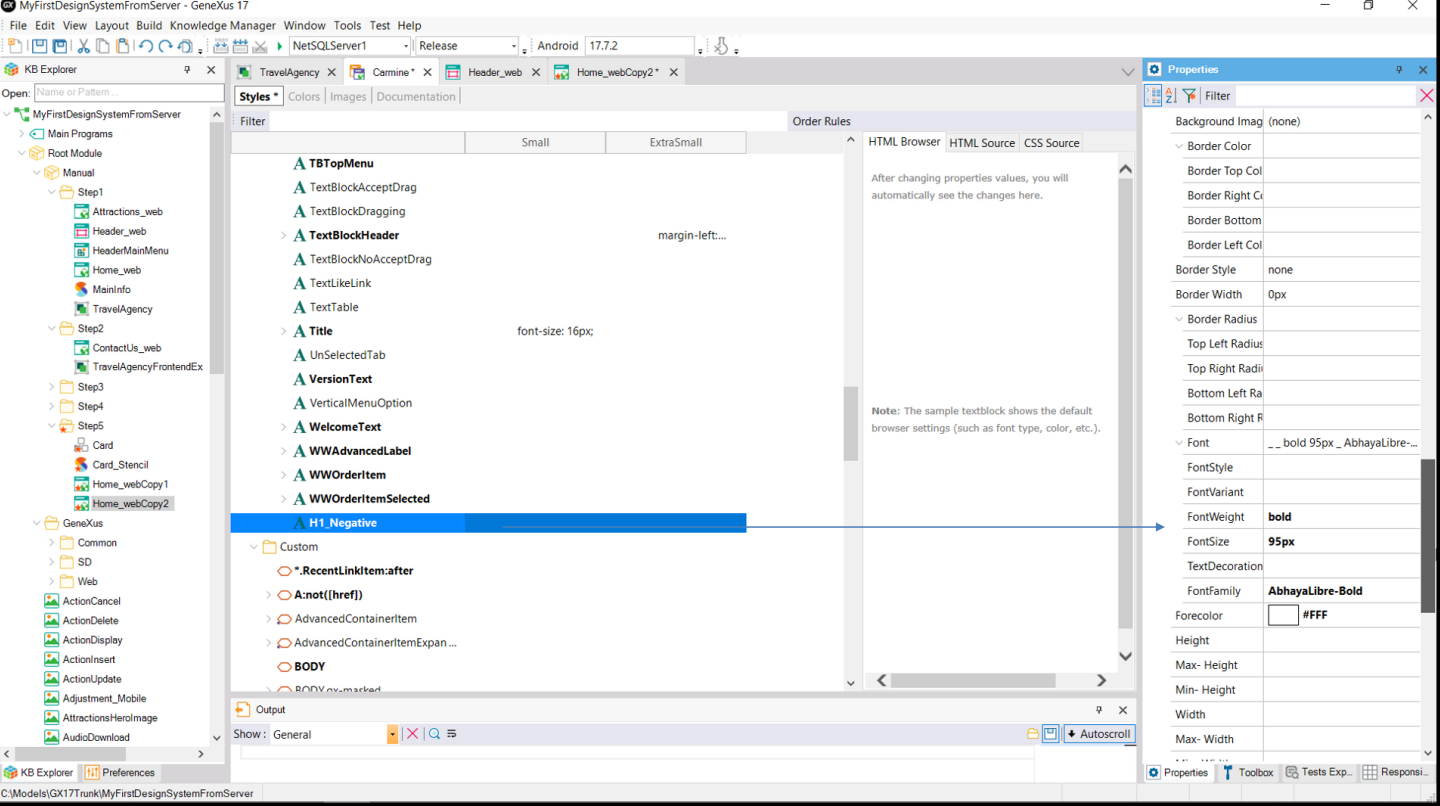
Properties Panel: Home_webCopy2

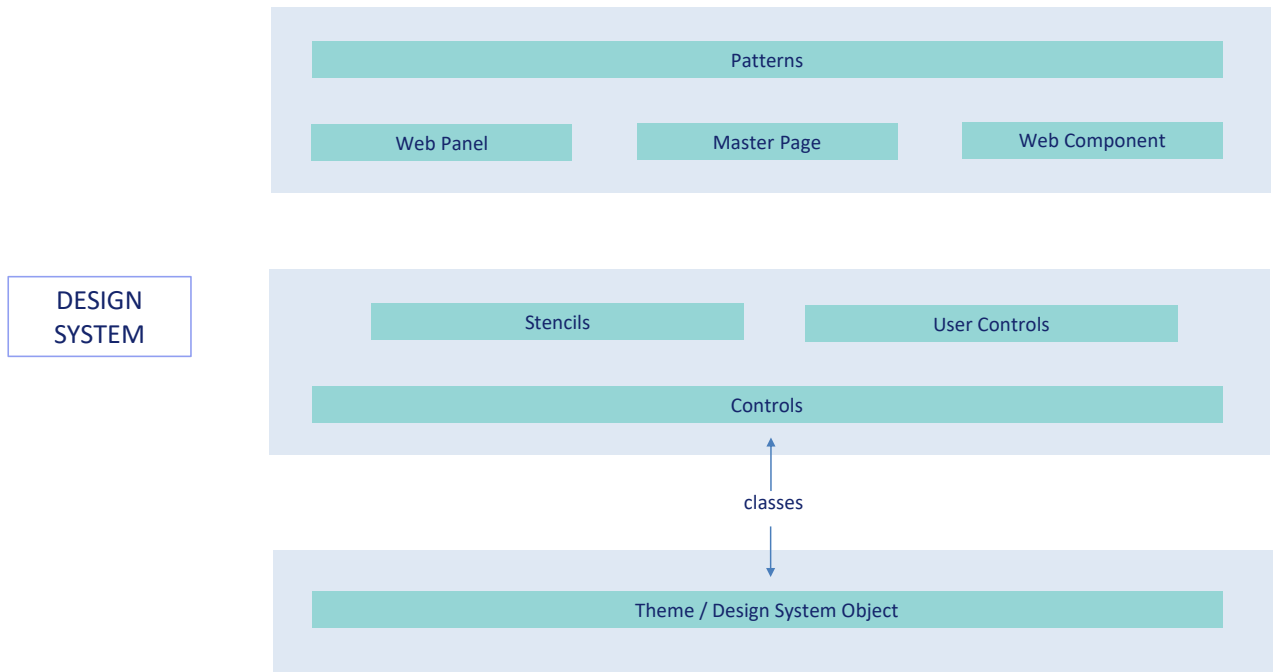
Name	Home_webCopy2
Description	Home_web_Copy2
Module/Folder	Step5
Style	Carmine
Type	Web Page
Master Page	Header_web
Show Master Page v	False
Main program	True
On session timeout	Ignore
Focus control	Use Environment property va...
Cache expiration lag	
Automatic refresh	Yes
Auto compress http	Use Environment property va...
Qualified Name	Manual.Home_webCopy2
Object Visibility	Public
Main object properties	
Web Application	Default
Location	
Generator	Default (C# Web)
Warning messages	
Disabled warning	spc0096 spc0107 spc0142
Compatibility	
Standard Function	Only standard functions
Web interface	
Web User Experie	Smooth

KB Explorer | Preferences

C:\Models\GX17Trunk\MyFirstDesignSystemFromServer







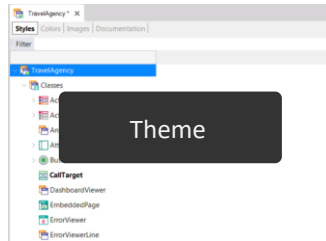
And this is how the controls are designed: through classes whose properties are defined in the Theme or Design System object.

So far, we have only discussed how to develop web applications. However, almost the same will apply to native mobile applications, or even Angular applications.

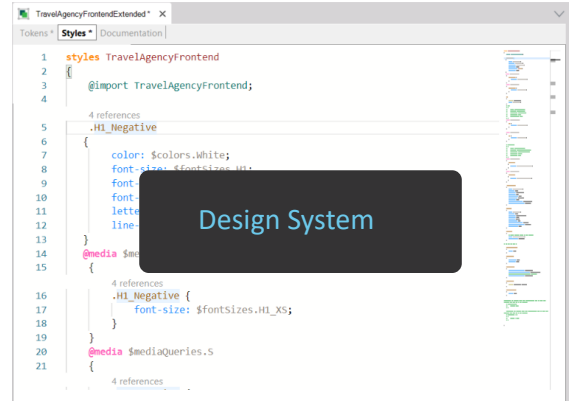
At the time this video was made, the latest upgrade of GeneXus 17 was number 7. There the default style object is still the Theme, but in future upgrades it will be changed to the Design System Object.

All this will become a reality and fully come to life when Unanimo—a complete Design System with a new design for the Pattern and transactions too—is released.

DESIGN
SYSTEM



Before



Now

Wiki: Design Systems (<https://wiki.genexus.com/commwiki/servlet/wiki?40108,Toc%3Design+Systems>)

There is a lot more to say and show, but here we simply wanted to present a complete overview of how the Design System of any application is modeled in GeneXus, at this turning point when we are changing from the use of the Theme object to the new Design System Object.

There is already enough material (in videos and wiki articles) for you to learn more about all this, whenever you want.